Rethinking Customer Segmentation and Demand Learning in the Presence of Sparse, Diverse, and Large-scale Data

by

Ashwin Venkataraman

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

New York University

September 2018

Professor Srikanth Jagabathula

Professor Lakshminarayanan Subramanian

Dedication

To my family, for their constant love, support and encouragement.

Acknowledgements

This dissertation is the product of close collaboration with my advisors, Lakshminarayanan Subramanian and Srikanth Jagabathula. I cannot begin to describe the influence and impact they have had on me, and am deeply indebted to them for their patience, guidance and support. They taught me how to do research, and instilled confidence in me to keep persevering despite tough times. I started working with Lakshmi immediately after I joined NYU and despite his innumerable other commitments, he always made time to discuss research and make sure everything was going fine. His amazing energy and passion will be a constant source of inspiration for me. He also introduced me to Srikanth in my first semester, which eventually resulted in him becoming my co-advisor and shaping much of my research interests. I am especially grateful to Srikanth for always being available to meet (including the many long meetings discussing proofs in his office!), despite being busy with numerous other projects. I hope that some day, I can be at least half as efficient as him in managing my time.

I would like to thank my committee, Prof. Kyunghyun Cho, Prof. Paat Rusmevichientong, and Prof. Jiawei Zhang, for providing many helpful comments and feedback that improved the work described in this dissertation.

I thank Prof. Yaw Nyarko and NYU Center for Technology and Economic Development (CTED) for funding me during my initial PhD years, as well as all the folks at NYU Abu Dhabi for being so warm and welcoming during each of my visits. I am also grateful to Prof. David Sontag for teaching the "Probabilistic Graphical Models" course, which introduced me to many of the fundamental concepts in machine learning, and to Prof. Jinyang Li, for helpful discussions and valuable advice in my first year.

I had a wonderful time at eBay NYC during two summer internships, and thank everyone there, especially my mentors, Yuri Brovman, Daniel Galron and Natraj Srinivasan, for their guidance and feedback. In particular, I am grateful to Yuri for coordinating with eBay's legal team, and allowing us to include the results obtained as part of my internships in our paper.

I thank all my co-authors and collaborators for their help and hard work. I am particularly grateful to Sunandan Chakraborty, for giving important career and life advice, and Rishabh Ranawat, for his enthusiasm and desire to learn. I thank Prof. Josh Reed and all the faculty in the IOMS department at NYU Stern for making me feel welcome and for many helpful interactions and feedback.

I thank the staff of the computer science department, especially Rosemary Amico and Santiago Pizzini, for all their assistance over the years. I also thank Leslie Cerve, our ever-present administrative aide, for taking care of the smallest things, and for the many wonderful and interesting conversations.

I am thankful for having wonderful colleagues and building new friendships during my time at NYU: Christopher Mitchell for patiently answering my queries and helping me with issues regarding the lab machines; Matt Tierney for providing advice when I just started (especially, on preparing for the Google interview which played a major part in me securing

an internship there); Vipin Jain for all the fun times in Abu Dhabi as well as in NY; Anantha Ravi Kiran, Ananth Balashankar, Cheng Tan, Chien-Chin Huang, Dmitry Mitrofanov, Edwin Reed-Sanchez, Fatima Zarinni, Huck Bennett, Jacopo Cirrone, Kate Boxer, Kunal Gugale, Mukund Sudarshan, Rishabh Ranawat, Renfei Zou, Ruchi Tandon, Sebastian Angel, Shankar Kalyanaraman, Shiva Iyer, Siddharth Krishna, Sunandan Chakraborty, Talal Ahmad, Tarek Abdallah, Trinabh Gupta, Varun Chandrasekaran, Yan Shvartzshnaider, Yang Zhang, and Zhaoguo (Tiger) Wang for many interesting discussions and conversations, both academic and otherwise. I am particularly thankful to Shiva, Talal and Varun, for their company and constant support, not to mention the numerous dinners and fun conversations. I am grateful to each one of them for providing an intellectually stimulating and cordial environment.

I am ever so grateful to Chaya and Seher for being the most amazing roommates one could hope for, and making our apartment a home away from home. Their support, care and understanding over the years made this journey so much more enjoyable (and bearable at times!), and I will always cherish all the wonderful times we spent together. I also thank my other roommates, Nadim and Akansha for putting up with my quirks and sharing both my happiness and disappointments.

Finally, I thank my family for their endless love, unwavering trust, and unconditional support.

Abstract

Firms are now able to collect unprecedented amounts of data. This wealth of data provides new opportunities and capabilities for the firm to better solve classical problems within operational and marketing contexts, such as customer segmentation and demand learning. At the same time, the data imposes new challenges. In addition to its *large-scale* nature which creates computational issues, the data comes from a *diversity* of sources, varying in their respective measurement scales (e.g., clicks, ratings, purchase signals, etc.), and is typically *sparse*, containing a large fraction of missing observations. The diversity in the data makes it hard to directly compare different observations (clicks vs purchases, for instance) and the severe sparsity precludes any meaningful imputations of unobserved entries. The data also comes from *unreliable* sources, which introduce both unintentional and deliberate errors. The identities of such sources is very often unknown, which makes it difficult to determine which sources to trust.

These data challenges require a *rethink* of traditional techniques for customer segmentation and demand learning. Given their importance and widespread use, this dissertation revisits the classical problems of customer segmentation and demand learning but in the presence of sparse, diverse, and large-scale data. The key contribution of the dissertation is a suite of novel methodologies to deal with the challenges described above.

Part I of the dissertation focuses on the problem of customer segmentation. In Chapter 1, we consider the problem of segmenting (or clustering) a large population of customers based on their preferences, when the preference signals (e.g., clicks, ratings, etc.) come from a multitude of diverse data sources and each customer provides only a few observations. These data characteristics preclude the applicability of traditional marketing techniques as well as standard clustering approaches in machine learning. We propose a *model-based embedding technique* which takes the customer observations and a probabilistic model class generating the observations as inputs, and outputs an *embedding*—a low-dimensional vector representation in Euclidean space–for each customer. We then cluster the embeddings to obtain the segments. We show that our segmentation technique can be used to generate highly accurate personalized recommendations in two real-world case studies, including upto 8% improvement over the existing approach on an eBay dataset consisting of millions of customers and items. In addition, it outperforms (both in speed and accuracy) standard techniques in marketing and machine learning.

In Chapter 2, we turn our attention to the domain of crowdsourced labeling, which provides a low-cost, easy and scalable way to collect labels from the crowd—composed of "workers"—which are then aggregated and used as inputs for training machine learning applications. The main challenge is that workers are often unreliable, and therefore can introduce unintentional or even intentional errors into the labels. The reliabilities of the workers are a priori unknown, so correctly aggregating the labels becomes difficult. We propose algorithms to separate the worker population into two segments, what we call "honest" and "adversarial" workers.

Honest workers can provide incorrect labels, but their errors are probabilistic and therefore, can be corrected. Adversarial workers, on the other hand, adopt arbitrary labeling strategies (whether deterministic or probabilistic) and therefore, their labels cannot be trusted. We demonstrate that discarding the labels provided by even a few adversarial workers can significantly improve the accuracy of several existing approaches for aggregating the labels in real-world crowdsourcing datasets.

Part II is devoted to demand learning. In Chapter 3, we consider the problem of learning customer demand for a set of substitutable products. Within operations, the customer demand is typically modeled using a mixture of logit models, which can capture heterogeneity as well as rich substitution patterns in customer preferences. The mixture model is fit to historical sales transactions and inventory data, and the fitted model is used to inform pricing and assortment decisions. We propose a novel nonparametric estimator for the mixture of logit models, providing the ability to make effective use of the large amounts of transaction data that firms have access to. By contrast, most existing techniques impose parametric assumptions—usually driven by tractability considerations—on the mixing distribution, and consequently can suffer from model misspecification issues. We show that our estimator is able to recover good approximations of different ground-truth mixing distributions—despite having no knowledge of their underlying structure—and outperforms the standard expectation-maximization (EM) benchmark in predictive and decision accuracies, while being an order of magnitude faster.

Table of Contents

List of Figures

List of Tables

List of Appendices

Preface

This dissertation describes a set of joint works with my advisors that have either already been published or are currently under review: Chapters 1 and 2 describe works published in [JSV18b] and [JSV17] respectively, while the work presented in Chapter 3 is currently under review at a journal [JSV18a]. As such, much of the content has been taken verbatim from the above papers, and it would be perfectly reasonable for the reader to choose to read them instead. The main differences include a global introduction (which appears below), additional discussion about practical algorithmic issues in Chapter 1, and streamlining of the proofs and details of numerical experiments in the appendices. There are also some small fixes to the notation and minor rewording throughout. Any errors introduced in this process are my own fault and not of my advisors.

Introduction

Firms are increasingly able to collect large amounts of data about their customers. Consider the example of eBay, an e-commerce platform allowing people to buy and sell merchandize. From the perspective of buyers,[1] eBay has access to the complete browsing activity of its customers—which products they have viewed in the past, along with fine-grained interactions like clicks, purchases, and product ratings/reviews that directly reveal their underlying preferences. It is able to collect such data via multiple channels like the eBay.com website and the eBay mobile app. In addition, it also has access to demographic information like age, gender, income, education, etc. This wealth of data provides opportunities for eBay to provide a more engaging and highly customized experience for its customers. For instance, eBay can leverage this data to predict what kinds of products customers will like to purchase and either recommend them directly, or with associated promotions/discounts to induce the customer to purchase. It can also create personalized home pages for the customers in order to increase engagement and eventually drive more number of purchases and higher revenues. Further, with growing computational capabilities, these decisions can be made both in real-time when the customer visits the website or mobile app, and offline which can then be sent in targeted e-mails, app notifications, etc.

However, the large amounts of data also poses several challenges in extracting the critical information needed to make such decisions. In order to highlight the concrete set of challenges, lets look at a specific task that eBay is interested in: recommending buyers "similar" products for purchase.[2] When a customer visits a product page on eBay, he/she is recommended "similar" products, which are shown below the viewed product. For instance, in Figure 1, the customer is currently viewing a pair of sneakers—called the "seed product"—and recommended below are other pairs of shoes on eBay which the customer might also be interested in purchasing—the exact same pair sold by a different seller (at a cheaper price), different color variants, another style within the same brand, and so on. eBay designed a scalable architecture and system to provide these recommendations in real-time, however,

[1] In a similar vein, eBay has information about sellers such as the set of products they have sold, their feedback scores, etc. The applications we consider are more from a buyer's viewpoint, and therefore we focus our discussion on the buyers.

[2] We discuss this problem in greater detail in Section 1.6.

Example of similar product recommendation on eBay

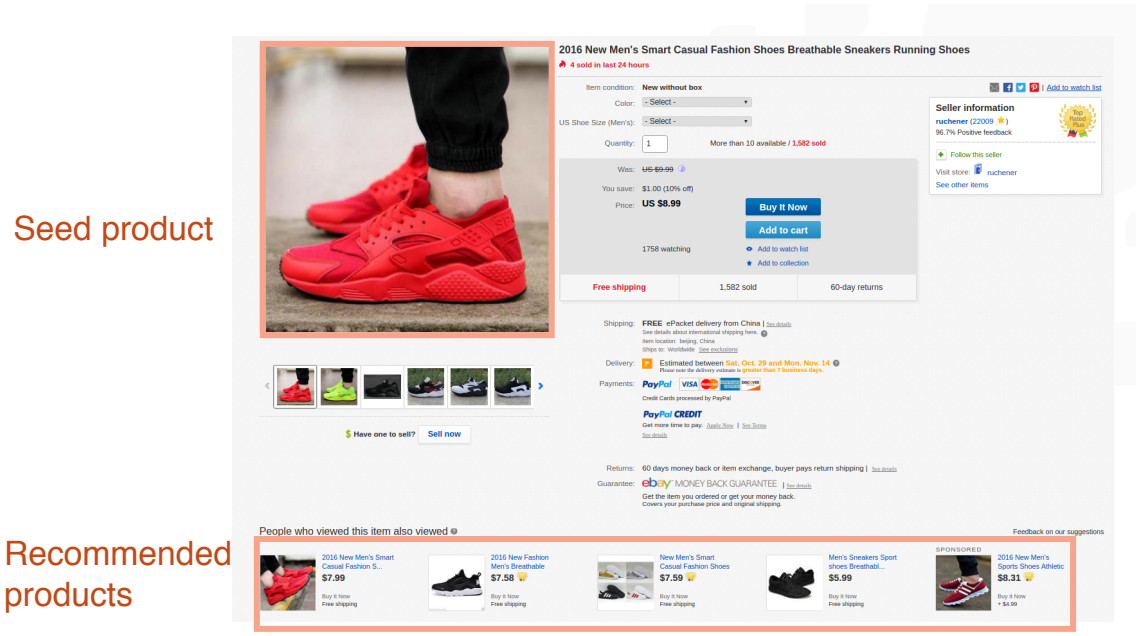

Figure 1: The seed product is a pair of sneakers that the user is currently viewing, and the recommended products at the bottom are other pairs of shoes similar to the seed product.

they would provide the same set of recommendations for a given viewed product, regardless of which customer was viewing it. Since customers typically have heterogeneous preferences for different product features—for instance, some customers might really like the red color of the sneakers and only want to see other red colored shoes, whereas other customers might be brand loyal and only want to be shown other styles in the same brand—eBay wanted to segment (or cluster) the customers according to their preferences, and personalize (or customize) the recommendations to each segment.

Despite the fact that eBay has access to large amounts of data about various interactions (such as clicks, purchases, etc.) from its customers, determining whether two customers actually have similar preferences and therefore, should be placed in the same segment, is extremely non-trivial. First, the *large-scale* nature of the data—eBay has millions of customers and millions of products in its catalog—imposes computational challenges, resulting in the need for clustering algorithms that can scale to such large data sizes. Second, the observed interactions come from *diverse* sources encompassing views, clicks, ratings, purchases, etc. and are represented on different measurement scales, which makes it difficult to directly compare different signals. If a customer provides a 5-star rating for a product and another customer purchases the same product, it is unclear if the two customers have similar preferences. Similarly, it is unclear whether a click provides the same customer intent as a purchase—people typically don't mind clicking on products just to explore, but are much more cautious when it comes to actually purchasing the product. Third, for an individual customer, there

are typically very few observations. In other words, the data is highly *sparse*. For instance, in our sample dataset consisting of customer interactions during a two-week period (see Section 1.6), a customer on average interacted with only 5 out of 4.2M products. As a result, two customers may have no overlap in the set of products they interact with, again making it challenging to determine if the customers have similar preferences. In addition, many of the products are *unstructured*, i.e. lacking well-defined or consistent feature representations, which limits the ability to extrapolate customer preferences. For example, eBay has a large collection of antique and collectible items in its catalog, which lack any reasonable feature structure. Finally, the preference observations can come from *unreliable* sources, which introduce both unintentional and intentional errors. For instance, fraudulent buyers can provide bad/negative feedback scores for sellers, despite having a smooth experience with their purchase.[3] Similarly, many of the products in eBay's catalog are associated with false or misleading description about their authenticity (replicas, counterfeits, etc.) and/or features.[4] Consequently, not all of the data can be trusted, and eBay needs mechanisms to filter out data from unreliable sources.

The above data characteristics also impose challenges on another classical problem faced by firms: learning the customer demand. For instance, retailers like Walmart, Target, Zara, and numerous others, want to be able to predict demand for different products that they offer to the population, and these demand predictions are used as inputs for key decisions like the assortment, i.e. set of products, to carry in a particular store and the prices to charge for the offered products, which products to recommend to new customers, introducing a new product to, or discontinuing an existing product from the market, and so on. To capture substitution behavior of customers between different products (if the product of choice is not available, or stocked-out, when the customer visits the store, he/she can instead purchase another "similar" product that is in stock), the standard approach in the operations and marketing literature is to specify a *choice model*—a probabilistic model that relates product attributes like price, brand, etc. and the set of offered products to the probability of purchase—and fit the model to historical sales transactions—which provide information on which products were purchased— and inventory data—which provide information on which other products were offered to the customer when a purchase was made—collected by the firm. The most popular choice model used in the literature is the mixture of logit model (also referred to as the mixed logit [RT98] model), which can capture heterogeneity as well as rich substitution patterns in customer preferences. It has been shown that the mixture of logit model can approximate a wide class of choice models [MT00] and it has been successfully applied in practice. However, a key challenge in fitting the mixture of logit model is that the structure of the mixing distribution—which models the distribution of preferences in the customer population—is not known in practice. Existing approaches typically impose parametric assumptions on the mixing distribution—usually driven by tractability considerations—which can result in model

[3] https://community.ebay.com/t5/Archive-The-Front-Porch/EBAY-IS-FILLED-WITH-FRAUDULENT-BUYERS/td-p/24847360

[4] https://www.ebay.com/help/policies/prohibited-restricted-items/fake-items-policy?id=4276

misspecification, i.e. the true distribution does not belong to the chosen parametric family. Model misspecification can result in biased estimates for parameters of interest [Tra08] as well as poor goodness-of-fit measures [FKRB11], leading to sub-optimal decisions. Further, these kind of issues are exacerbated when firms have access to large amounts of transaction data, often coming from different geographic and/or demographic regions, resulting in highly diverse and dynamic customer choice behavior that need not necessarily be explained by distributions belonging to the same parametric family. In particular, parametric estimators do not scale with the amount of data or "information" available and therefore, are not able to make effective use of the large-scale transaction data that firms have access to nowadays.

The aforementioned data challenges require a *rethink* of traditional techniques for customer segmentation and demand learning. Given their importance and widespread use, this dissertation revisits the classical problems of customer segmentation and demand learning but in the presence of sparse, diverse, and large-scale data. The key contribution of the dissertation is a suite of novel methodologies to deal with the challenges described above.

Our Contributions

This dissertation contains two parts: In Part I, we study the problem of customer segmentation, and Part II is devoted to learning customer demand. Under customer segmentation, we focus on two problems: (i) segmenting customers based on their preferences, where we address the challenges of diversity, sparsity and scale in the observed preference signals and (ii) segmenting crowd workers based on their reliability, where we deal with the issue of unreliability in the observed data. Under demand learning, we consider the problem of estimating a mixture of logit models from large-scale historical sales transactions and inventory data.

Segmenting customers based on their preferences

In Chapter 1, we consider the problem of segmenting (aka clustering) a large population of customers based on their preferences over a large collection of items, when the preference signals (e.g., clicks, purchases, ratings, etc.) come from a multitude of diverse data sources and each customer may provide only a few observations. Further, the items may lack any reasonable feature structure, making it difficult to extrapolate customer preferences. These data characteristics limit the applicability of existing techniques in marketing and machine learning. To overcome these limitations, we propose a *model-based embedding technique* which takes the customer observations and a probabilistic model class generating the observations as inputs, and outputs an *embedding*—a low-dimensional vector representation in Euclidean space—for each customer. We then cluster the embeddings (using a standard clustering technique such as k-means) to obtain the segments.

The key novelty of our approach is generating the embedding, which leverages the probabilistic model to convert a categorical observation like click or purchase into its corresponding (log-)likelihood value under the model. We estimate the model parameters by pooling together

the data from all customers and ignoring the possibility that different customers may have different model parameters. This results in a model that describes a 'pooled' customer—a virtual customer whose preferences reflect the aggregated preferences of the population. The likelihood transformations then measure how much a particular customer's preferences differ from those of the population's. Our theoretical analysis shows that under reasonable assumptions, customers from different segments will have different (log-)likelihood values under the pooled model—allowing us to separate them out.

We outline our contributions below:

1. *Novel segmentation algorithm.* Our algorithm is designed to operate on large customer populations and large collections of (unstructured) items. Moreover, it is (a) *principled*, reducing to standard algorithms in machine learning in special cases; (b) *fast*, with an order of magnitude speedup compared to benchmark latent class models because it requires fitting only one model (as opposed to a mixture model); and (c) *flexible*, allowing practitioners to systematically incorporate problem-dependent structures through the probabilistic model class, providing a way to take advantage of the rich literature in marketing proposing models for individual customer behavior. We refer the reader to Section 1.2 for the detailed description of the algorithm.

2. *Analytical results.* Under a standard latent class model for customer observations, we derive necessary and sufficient conditions for asymptotic recovery of the true segments. Specifically, we bound the asymptotic *misclassification rate*, defined as the expected fraction of customers incorrectly classified, of a nearest-neighbor classifier trained on customer embeddings obtained from the embed step in our algorithm. Given a universe of n items such that each customer provides at least $\log n$ observations, we show that the misclassification rate scales as $O\left(n^{-\frac{2\Lambda^2\alpha_{\min}^2}{81}}\right)$ where $0 < \alpha_{\min}, \Lambda < 1$ are constants that depend on the underlying parameters of the model. In other words, when each customer provides $O(\log n)$ observations, our algorithm correctly classifies *all* customers into their respective segments, as $n \to \infty$. The formal statements of the results are presented in Section 1.3. Our results are similar in spirit to the conditions derived in existing literature for Gaussian mixture models [AM05, KSV05].

3. *Empirical results.* We conducted three numerical studies to validate our methodology:

   (a) Using synthetic data (see Section 1.4), we show that our method obtains more accurate segments, while being upto 17× faster, than the standard latent class (LC) benchmark.

   (b) On the publicly available `MovieLens` dataset [HKBR99], we apply our segmentation method to solve the classical *cold-start problem* in recommender systems, specifically, the problem of recommending new movies to customers. We show that segmenting customers using our method and customizing recommendations to each segment improves the recommendation accuracy by 48%, 55%, and 84%

for movies in drama, comedy, and action genres, respectively, when compared to a baseline method that treats all customers as having homogeneous preferences. It also outperforms the standard LC (by upto 13%) and empirical bayesian (by upto 19%) benchmarks used for capturing heterogeneity in the marketing literature, as well as common clustering techniques in the machine learning literature. Refer to Section 1.5 for the details.

(c) On a real-world dataset from eBay (see Section 1.6), we apply our segmentation methodology for personalizing similar product recommendations. We show that segmenting the population using our approach and customizing recommendations to each segment can result in upto 8% improvement in the recommendation accuracy, when compared to treating the population as having homogeneous preferences. The improvement of 8% is non-trivial because before our method, eBay tried several natural ways to segment (by similarity of demographics, frequency/recency of purchases, etc.), but the best of them resulted in $\sim 1\%$ improvement.

Segmenting crowd workers based on their reliability

In Chapter 2, we consider the problem of segmentation in the presence of unreliable data. In particular, we focus on the domain of crowdsourced labeling, which provides a low-cost, easy and scalable way to collect labels from the crowd—composed of "workers"—which are then aggregated and used as inputs for training machine learning applications. The key challenge here is that workers are often unreliable, and therefore can introduce unintentional or even intentional errors into the labels. The reliabilities of the workers are a priori unknown, so correctly aggregating the labels becomes difficult. Most existing studies assume that the worker labels are generated according to specific probabilistic models; however, recent evidence shows the presence of workers adopting non-random or even malicious strategies. To account for such workers, we suppose that the worker population comprises a mixture of, what we call, *honest* and *adversarial* workers. Honest workers may be reliable or unreliable (i.e., they can also provide incorrect labels), but they provide labels according to an unknown but explicit probabilistic model. Adversarial workers, on the other hand, adopt labeling strategies different from those of honest workers, whether probabilistic or not. However, the identities (honest or adversarial) of individual workers are unknown. We propose two reputation algorithms (that only use the observed worker labels) to separate out unreliable honest workers and adversarial workers in the population. Our algorithms assume that honest workers are in the majority, and they classify workers with outlier label patterns as adversaries.

Specifically, we make the following contributions:

1. *Reputation algorithms.* We propose a reputation-based algorithm to identify outlier worker labeling patterns. The algorithm takes as inputs a set of workers, a set of labeling tasks having (unknown) true labels in $\{-1, +1\}$, and binary ($+1$ and -1) labels provided by the workers. Each worker may label only a subset of the tasks.

Our algorithm makes no specific assumptions on the worker labeling strategies and identifies outliers by *penalizing* workers for the number of "conflicts" they are involved in. Specifically, suppose each task receives both $+1$ and -1 labels from the workers. For each task t, the algorithm maintains the number d^+ of $+1$ labels, number d^- of -1 labels, and a penalty budget of 2. Intuitively, if $d^+ > d^-$, then the worker assigning label -1 to task t is "more" of an outlier than a worker assigning label $+1$. Accordingly, the algorithm makes the following decisions: (a) how much of the penalty budget to allocate to each worker labeling a given task, and (b) how to aggregate the penalties allocated to each worker (across all tasks assigned to the worker) to arrive at the final reputation score. We outline two algorithms that differ in how they make these two decisions: *soft-penalty* and *hard-penalty*. Both penalty assignments are based on the intuition that when there are more honest workers than adversaries in the population, the former are more likely to agree with the majority label on a given task and therefore receive lower penalties. Refer to Section 2.3 for the formal description of the algorithms.

2. *Analytical results.* We analyze our reputation algorithms under three settings: (a) there are no adversaries in the population and honest workers adopt the standard "one-coin" model, which assumes that each worker w provides the correct label to an assigned task with probability p_w and the incorrect label with probability $1 - p_w$ (the parameter p_w thus measures the *reliability* of worker w), to generate the labels; (b) honest workers adopt the one-coin model and adversaries, the Uniform strategy in which they label all assigned tasks $+1$; and (c) honest worker labels are generated according to the spammer-hammer model [KOS11] and the adversaries are *sophisticated*, having infinite computational capacity as well as knowledge of the honest workers' labels, and therefore capable of executing arbitrary labeling strategies.

   For the first two settings, we consider a standard crowdsourced labeling setup in which each worker labels the same number l of tasks and each task is assigned to r workers. For the first setting, we show that the penalties assigned by the soft-penalty algorithm are consistent with worker reliabilities in the one-coin model—the higher the reliability, the lower is the penalty. For the second setting, we derive necessary and sufficient conditions under which the soft-penalty algorithm assigns lower (expected) penalties to honest workers than to adversaries, and bound the asymptotic misclassification rate of a threshold classifier—given a penalty threshold θ, the threshold classifier classifies a worker as honest if its penalty is less than or equal to θ and as adversarial otherwise—showing that it correctly classifies all adversaries and honest workers with above average reliabilities.

   For the last setting, we make no assumptions on the structure of the task assignment to the workers. This setup is reflective of public crowdsourced settings (such as Amazon, Yelp, etc.) in which workers choose which tasks to label and the assumption of sophisticated adversaries is most relevant, and we derive performance guarantees based on the resulting assignment structure. We show that the hard-penalty algorithm is robust to sophisticated adversary strategies but the soft-penalty algorithm is vulnerable.

In particular, we propose an aggregation algorithm that utilizes the worker penalties computed by the hard-penalty algorithm, and establish the existence of an upper bound on the worst-case number of tasks whose true label is incorrectly inferred by this aggregation algorithm, under *any* adversary strategy. See Section 2.4 for the detailed description of the results.

3. *Numerical results.* We conducted two numerical studies to demonstrate the practical value of our reputation algorithms. The first study shows, on five real-world crowdsourcing datasets, that *discarding* the labels provided by adversaries identified by our algorithms allows *all* of the following standard label aggregation algorithms to infer true task labels more accurately: (a) simple majority, (b) expectation-maximization (EM) for the two-coin model [RY12], (c) KOS [KOS11], (d) a normalized variant of KOS, (e) spectral EM [ZCZJ14], and (f) regularized minimax conditional entropy [ZLP+15]. In particular, we show that by filtering out even a few workers (upto at most 10), our methods can improve the accuracy of the inferred task labels by 9.8% on average. These improvements suggest that the label patterns of discarded workers do not conform to standard probabilistic models assumed in prior works. Refer to Section 2.5.1 for the details.

   The second study (see Section 2.5.2) is designed to complement our theoretical analysis. By using synthetic data, we show that both the soft- and hard-penalty algorithms successfully identify Uniform adversaries (who label all assigned tasks +1) and low-reliability honest workers, when the number of tasks a worker labels has a power-law distribution. This scenario commonly arises in crowdsourcing settings where workers organically choose which tasks to label, with some workers labeling many tasks and some tasks receiving many labels [FKK+11]. The study also offers insights into the settings under which the soft- or hard-penalty algorithm is appropriate; we observe that the soft-penalty (hard-penalty) algorithm is more appropriate when the number of tasks that an adversary labels is low (high).

Nonparametric estimator for mixture of logit models

Mixture of logit models are commonly used for modeling customer demand in econometrics, operations and marketing. A key challenge in estimating the mixture model is that the structure of the mixing distribution is often unknown in practice. Most existing techniques impose parametric assumptions—usually driven by tractability considerations—on the mixing distribution, and consequently can suffer from model misspecification issues. These issues are exacerbated when firms have access to large amounts of data, reflecting highly diverse and dynamic choice behavior from different subpopulations of customers, since parametric estimators do not scale with the amount of data that is available. This motivates the need for designing nonparametric estimators that can automatically adapt and learn the customer choice behavior from the observed data.

In Chapter 3, we propose a novel nonparametric method for estimating a mixture of

logit models from large-scale historical sales transactions, and inventory data, which provides information on product availability. Together, these data specify the number of sales for each product as a function of the offer-set, i.e. the subset of products offered to the customer population, for a collection of offer-sets. Our estimation technique finds the best fitting distribution to the data, where the fit is measured through a convex loss function, amongst the class of *all* possible mixing distributions. We formulate the estimation problem as a constrained convex program and leverage the conditional gradient (aka Frank-Wolfe) algorithm to solve it, showing that it iteratively recovers the support of the mixing distribution. The recovered mixing distribution is discrete, with each component in the support representing a particular 'customer type'. The conditional gradient (CG) algorithm has received a lot of attention in the machine learning literature recently, with many variants being proposed for solving large-scale optimization problems, and our estimation method can also leverage these developments for effectively learning the customer demand from large-scale transaction data.

We summarize our key contributions below:

1. *Novel mixture estimation methodology.* Our estimator is (a) *general-purpose*: can be applied with little to no customization for a broad class of loss functions; (b) *fast*: order of magnitude faster than the benchmark EM algorithm; and (c) *nonparametric*: makes no assumption on the mixing distribution and estimates customer types in the population in a data-driven fashion. We refer the reader to Section 3.2 for the problem setup and detailed formulation of our approach.

2. *Analytical results.* We obtain two key theoretical results:

   (i) We provide convergence guarantees for our CG-based estimator, for both the log-likelihood and squared loss functions (see Theorem 3.2 in Section 3.4.1). For the squared loss, we obtain a sublinear convergence rate of $O(1/k)$ after k iterations, which follows directly from applying existing results in our context. But, for the log-likelihood loss, existing results don't apply because the gradient of the loss function blows up at the boundary of the constraint region of the convex program. We address this issue by showing that the iterates produced by the fully corrective variant of the CG algorithm (the one that we implement) are strictly bounded away from the boundary. We then adapt and extend existing arguments to establish the same sublinear convergence guarantee, as for the squared loss.

   (ii) We characterize the structure of the mixing distribution recovered by our estimator. Our method recovers two kinds of customer types: what we call, (a) non-boundary and (b) boundary types. A non-boundary type is described by a standard logit model with a parameter vector $\boldsymbol{\omega}$—see Section 3.2 for the precise form of the logit model we consider. The boundary types, on the other hand, are limiting logit models that result from unbounded solutions in which the parameter vector $\boldsymbol{\omega}$ is pushed to infinity. We show that each boundary type can be described by two parameters $(\boldsymbol{\omega}_0, \boldsymbol{\theta})$. The parameter vector $\boldsymbol{\theta}$ induces a (weak) preference order over the set of products and determines a consideration set the customer forms,

when given an offer-set. The parameter vector $\boldsymbol{\omega}_0$ then determines the logit choice probabilities from within the consideration set. The detailed results are presented in Sections 3.4.2 and 3.4.3.

3. *Empirical results.* We conducted three numerical studies to validate our methodology:

   (a) Using synthetic data (see Section 3.5), we show that our estimator is robust to several complex ground-truth mixing distributions and consistently recovers a good approximation to the underlying distribution. This is despite our estimator having no knowledge of the structure of the true mixing distribution. In particular, its performance is significantly better than a standard benchmark method imposing a parametric assumption on the mixing distribution, and highlights the potential impact of model misspecification in practice.

   (b) On the SUSHI Preference Dataset [KKA05] consisting of preference orderings given by customers for different sushi varieties, we show that our method achieves superior in-sample loss as compared to a latent class MNL (LC-MNL) model [Bha97] fit using the EM algorithm—while using the same (or even fewer) number of customer types—for both the log-likelihood (24% better) and squared (58% better) loss functions, with 16× speedup in the estimation time. The CG algorithm iteratively adds customer types that explain the observed choice data to the mixing distribution, which results in a much better fit as compared to the EM algorithm that updates all customer types together in each iteration. Our approach also achieves better predictive accuracy than EM, with an average 28% and 16% reduction in the RMSE (root mean square error) and MAPE (mean absolute percentage error) metrics for predicting market shares on new assortments.

   We also use the fitted mixture model to solve the assortment optimization decision, i.e. determining the subset of products to offer to the population that maximizes revenue, and show that our estimation technique can extract around 23% more revenue from the population, compared to the EM benchmark. This is an important finding because note that we are fitting the exact same model (mixture of logit) in both scenarios and are only changing the estimation technique, and highlights the amount of money that a sub-optimal method can leave on the table. Refer to Section 3.6.1 for the precise details.

   (c) On the IRI Academic Dataset [BKM08] which contains real-world sales transaction data from different grocery stores, we show that our method achieves upto 8% and 7% improvement in the in-sample log-likelihood and squared loss respectively, compared to the EM benchmark. Similarly, we achieve upto 7% and 5% reduction in out-of-sample log-likelihood and squared loss, respectively. In particular, our method outperformed EM-based estimation in all 5 product categories—shampoo, toothbrush, household cleaner, yogurt, and coffee—that we focused on. Section 3.6.2 reports the detailed results.

Part I

Customer Segmentation

Chapter 1

Segmenting customers based on their preferences

1.1 Introduction

'Customer segmentation' is the practice of grouping customers into non-overlapping segments (or clusters[1]) such that customers in the same segment have similar needs and preferences. It is now a ubiquitous practice carried out by firms. It allows them to effectively customize their product offerings, promotions, and recommendations to the particular preferences of each segment [Smi56]. Segmentation also subsumes *personalization* as a special case by placing each customer into a separate segment of her own. Personalization has gained a lot of traction recently. Yet, in most settings, customizing offerings to coarser segments is more meaningful than personalizing to individual customers simply because firms lack sufficient data for each customer. For example, in the sample dataset we use for our case study (in Section 1.6) on eBay, we see that customers often interact with less than 5 items, of eBay's massive online catalog consisting of more than 4M items.

The biggest challenge to carrying out segmentation is precisely this data sparsity. This challenge has become even more severe with firms being able to collect increasingly fine-grained observations such as direct purchases, ratings, and clicks, in addition to any demographic data such as age, gender, income, etc. These data are not only "big" (consisting of millions of customers and items), but also "complicated" in that they are (a) *diverse*, including actions that are represented on different scales (a click is not the same as a purchase is not the same as a rating), (b) *highly sparse*, spanning only a small fraction of the entire item universe, and (c) *unstructured*, with the items lacking well-defined feature information. Going back to the example above, eBay has a large and diverse product catalog consisting of products ranging from a Fitbit tracker or iPhone (products with well-defined attributes) to obscure antiques and collectibles (that lack any reasonable feature structure). Of these, each customer may click, purchase, or rate only a few items.

[1] We use the terms "segmentation" and "clustering" interchangeably.

In this chapter, we revisit the problem of segmentation but in the context of "big" and "complicated" data. These data characteristics pose new and unique challenges. First, traditional techniques within marketing don't apply. They assume that both customers and items have well-defined and consistent feature representations and often analyze small samples of customer populations. But, when the item universe is large and unstructured, customers can only be represented as large vectors with millions of entries, where each entry captures an action (say, purchase) taken on an item. These representations 'as is' are often meaningless for the purposes of segmentation. Almost all of their entries are missing and the lack of consistent feature representations of items means that missing entries can't be meaningfully imputed—for instance, a customer's purchase of an iPhone may reveal nothing about her propensity to purchase a particular antique. Existing techniques also become computationally intractable when classifying large populations of customers into segments. Second, the diversity of the types of actions captured in the vectors and their incompleteness make it difficult to assess similarity of customers. If a customer has clicked an item but another has purchased it, are they similar? How about customers who have purchased completely different subsets of items? This difficulty in obtaining a meaningful similarity measure precludes the application of standard clustering techniques in machine learning, despite being able to scale to large datasets.

To overcome the above challenges, we propose a *model-based embedding technique* that extends extant clustering techniques in machine learning to handle categorical observations from diverse data sources and having (many) missing entries. We focus on the setting where the objective of segmentation is to improve the performance on a prediction task. The precise prediction task depends on the application at hand, and includes predicting the probability of a customer clicking, purchasing, or liking an item. The algorithm takes as inputs the observations from a large population of customers and a probabilistic model class describing how the observations are generated from an individual customer. The choice of the model class is determined by the corresponding prediction task, as described below, and provides a systematic way to incorporate domain knowledge by leveraging the existing literature in marketing, which has proposed rich models describing individual customer behavior. It outputs an *embedding* for each customer—a vector representation in a low-dimensional Euclidean space whose dimension is much smaller than the number of items in the universe. The vector representations are then clustered, using a standard technique such as k-means, spectral clustering, mean-shift clustering, etc., to obtain the corresponding segments.

Put together, the algorithm proceeds in two sequential steps: *embed* and *cluster*. The *embed* step addresses the issue of diversity of the observed signals by first transforming the categorical observations into a continuous scale that makes different observations (such as purchases and ratings) comparable. It then deals with the issue of missing data by projecting the transformed observations onto a low-dimensional space, to obtain a vector representation for each customer. The *cluster* step then clusters the resulting vector representations to obtain the segments.

The key novelty of our algorithm is the *embed* step, which uses a probabilistic model to convert a categorical observation into its corresponding (log-)likelihood value under the model.

For example, if a customer likes an item with probability $\alpha \in [0, 1]$, then a "like" observation is transformed into $\log \alpha$ and "dislike" observation is transformed into $\log(1 - \alpha)$. We call our algorithm *model-based* because it relies on a probabilistic model; Section 1.5 presents a case study in which we illustrate the choice of the model when the objective is to accurately predict if a new movie will be liked by a customer. We estimate the model parameters by pooling together the data from all customers and ignoring the possibility that different customers may have different model parameters. This results in a model that describes a 'pooled' customer—a virtual customer whose preferences reflect the aggregated preferences of the population. The likelihood transformations then measure how much a particular customer's preferences differ from those of the population's. The theoretical analysis in Section 1.3 shows that under reasonable assumptions, customers from different segments will have different (log-)likelihood values under the pooled model—allowing us to separate them out.

Our algorithm is inspired by existing ideas for clustering in the theoretical computer science literature. It also systematically generalizes algorithms that are popular within the machine learning community. Particularly, when customer observations are continuous-valued, a model is not necessarily needed to transform them into a comparable scale. Then, when there are no missing entries, our segmentation algorithm with the appropriate cluster step reduces to the standard spectral projection technique [AM05, KSV05] for clustering real-valued observations. When there are missing entries, the embed step reduces to matrix factorization, which is commonly used in collaborative filtering applications [KBV+09].

1.1.1 Relevant literature

Our work has connections to literature in both marketing and machine learning.

Marketing. Customer segmentation is a classical marketing problem, with origins dating back to the work of [Smi56]. Marketers classify various segmentation techniques into *a priori* versus *post-hoc* and *descriptive* versus *predictive* methods, giving rise to a 2×2 classification matrix of these techniques [WK00]. Our algorithm is closest to the post-hoc predictive methods, which identify customer segments on the basis of the estimated relationship between a dependent variable and a set of predictors. The traditional method for predictive clustering is automatic interaction detection (AID), which splits the customer population into non-overlapping groups that differ maximally according to a dependent variable, such as purchase behavior, on the basis of a set of independent variables, like socioeconomic and demographic characteristics [Ass70, MJ81]. However, these approaches typically require large sample sizes to achieve satisfactory results. [Oga87] and [Kam88] proposed hierarchical segmentation techniques tailored to conjoint analysis, which group customers such that the accuracy with which preferences/choices are predicted from product attributes or profiles is maximized. These methods estimate parameters at the individual-level, and therefore are restricted by the number of observations available for each customer. Clusterwise regression methods [WK89, WS89] overcome this limitation, as they cluster customers such that the regression fit is optimized within each cluster.

Latent class (or mixture) methods offer a statistical approach to the segmentation problem,

and belong to two types: mixture regression and mixture multidimensional scaling models. Mixture regression models [WD94] simultaneously group subjects into unobserved segments and estimate a regression model within each segment, and were pioneered by [KR89] who propose a clusterwise logit model to segment households based on brand preferences and price sensitivities. This was extended by [GC94] who incorporated demographic variables and [KKL96] who incorporated differences in customer choice-making processes, resulting in models that produce identifiable and actionable segments. Mixture multidimensional scaling (MDS) models [DMM94] simultaneously estimate market segments as well as preference structures of customers in each segment, for instance, a brand map depicting the positions of the different brands on a set of unobserved dimensions assumed to influence perceptual or preference judgments of customers.

The purpose of the above model-based approaches to segmenting customers is fundamentally different from our approach. These methods focus on characterizing the market segments in terms of product and customer features (such as prices, brands, demographics, etc.) by analyzing structured products (i.e. having well-defined attributes) and small samples of customer populations; consequently, they do not scale to directly classifying a large population of customers. Our algorithm is explicitly designed to classify the entire customer population into segments, and can be applied even when the data is less-structured or unstructured (refer to the case study in Section 1.6). Another distinction is that we can provide necessary and sufficient conditions under which our algorithm *guarantees* asymptotic recovery of the true segments under a latent class model, which is unlike most prior work. In addition, our algorithm can still incorporate domain knowledge by leveraging the rich models describing customer behavior proposed in existing literature.

Machine Learning. Clustering is defined as the problem of partitioning data objects into groups, such that objects in the same group are similar, while objects in different groups are dissimilar. The literature on clustering is vast; see [XW05] and [Jai10] for excellent reviews. The most popular type of clustering approaches specify a distance/similarity measure between data points and determine the segments by optimizing a merit function that captures the "quality" of any given clustering. The popular k-means (and its variants k-medians, k-medoids, etc.), hierarchical clustering [RM05], and spectral clustering [SM00, NJW01] are notable examples. However, as mentioned earlier, the diversity and sparsity in observations make it challenging to construct a meaningful similarity measure and limit the direct applicability of such techniques for clustering the customer population. At the same time, any of these techniques can be used in the *cluster* step of our algorithm (see Section 1.2.3), which enables us to tap into this vast literature. In contrast to the above similarity-based clustering approaches, model-based clustering techniques [FR02, ZG03] assume that each cluster is associated with an underlying probabilistic model and different clusters differ on the parameters describing the model. They estimate a finite mixture model [MP00] to the data and classify customers based on the posterior membership probabilities. Our segmentation approach is closer to these techniques. The key distinction however is that these techniques estimate the parameters for each mixture component and can suffer when the number of samples available is limited, resulting in inaccurate segment classifications (see Section 1.4). Our approach, on the other

hand, fits only a single model, the pooled model, and therefore is more robust to sparsity in the customer observations and at the same time, achieves significant computational speedup.

Our work also has methodological connections to work in the theoretical computer science literature on learning mixture models. Specifically, our model-based embedding technique extends existing techniques for clustering real-valued observations with no missing entries [AM05, KSV05] to handle diverse categorical observations having (many) missing entries. Finally, method-of-moment based techniques with strong theoretical guarantees have recently been proposed for learning specific mixture models [HK13, AGH+14]. However, these approaches require the lower-order moments to possess specific structures, which do not hold in our setting.

1.2 Setup and Algorithmic Framework

Our goal is to segment a population $[m] \stackrel{\text{def}}{=} \{1, 2, \ldots, m\}$ of m customers comprised of a fixed but unknown number K of non-overlapping segments. To carry out the segmentation, we assume access to individual-level observations that capture differences among the segments. The observations may come from diverse sources—organically generated clicks or purchases during online customer visits; ratings provided on review websites (such as Yelp, TripAdvisor, etc.) or recommendation systems (such as Amazon, Netflix, etc.); purchase attitudes and preferences collected from a conjoint study; and demographics such as age, gender, income, education, etc. Such data are routinely collected by firms as customers interact through various touch points. Without loss of generality, we assume that all the observations are categorical—any continuous observations may be appropriately quantized. The data sources may be coarsely curated based on the specific application but we don't assume access to fine-grained feature information.

To deal with observations from diverse sources, we consider a unified representation where each observation is mapped to a categorical label for a particular "item" belonging to the universe $[n] \stackrel{\text{def}}{=} \{1, 2, \ldots, n\}$ of all items. We use the term "item" generically to mean different entities in different contexts. For example, when observations are product purchases, the items are products and the labels binary purchase/no-purchase signals. When observations are choices from a collection of offer sets (such as those collected in a choice-based conjoint study), the items are offer sets and the labels are IDs of chosen products. Finally, when observations are ratings for movies, the items are movies and the labels star ratings. Therefore, our representation provides a compact and general way to capture diverse signals. We index a typical customer by i, item by j, and segment by k.

In practice, we observe labels for only a small subset of the items for each customer. Because the numbers of observations can widely differ across customers, we represent the observed labels using an edge-labeled bipartite graph $\mathcal{P}$, defined between the customers and the items. An edge (i, j) denotes that we have observed a label from customer i for item j, with the edge-label x_{ij} representing the observed label. We call this graph the *customer-item preference graph*. We let $\boldsymbol{x}_i$ denote the vector of observations for customer i with $x_{ij} = \phi$

if the label for item j from customer i is unobserved/missing. Let $N(i)$ denote the set of items for which we have observations for customer i. It follows from our definitions that $N(i)$ also denotes the set of neighbors of the customer node i in the bipartite graph $\mathcal{P}$ and the degree $d_i \stackrel{\text{def}}{=} |N(i)|$, the size of the set $N(i)$, denotes the number of observations for customer i. Note that the observations for each customer are typically highly incomplete; therefore, $d_i \ll n$ and the bipartite graph $\mathcal{P}$ is highly sparse.

We assume that a customer's observations are generated according to an underlying parametric model from a pre-specified model class $\mathcal{F}(\Omega) = \{f(\boldsymbol{x}; \omega) : \omega \in \Omega\}$, where Ω is the space of latent preference parameters, $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \mathcal{B} := \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$ is the vector of item labels, and $f(\boldsymbol{x}; \omega)$ is the probability of observing the item labels $\boldsymbol{x}$ from a customer with model parameter ω. Here, $\mathcal{X}_j$ is the domain of possible categorical labels for item j. When all the labels are binary, $\mathcal{X}_j = \{0, 1\}$ and $\mathcal{B} = \{0, 1\}^n$. The choice of the parametric model class depends on the application context and the prediction task at hand. For instance, for the task of predicting whether a customer likes a movie or not, $\mathcal{F}$ can be chosen to be the binary logit model class; for the task of predicting movie ratings (say, on a 5-star scale), $\mathcal{F}$ can be the ordered logit model class; and for the task of predicting which item will be purchased, $\mathcal{F}$ can be the multinomial logit (MNL) model class. Depending on the application, other models proposed within the marketing literature may be used. We provide concrete illustrations as part of case studies in the `MovieLens` dataset (Section 1.5) and eBay dataset (Section 1.6).

In order to segment the customer population, we assume that the population is heterogeneous, comprising different segments. Each segment is distinguished by its latent preference parameters, so that a population consisting of K segments is described by K distinct models $f_1, f_2, \cdots, f_K$ with corresponding latent preference parameters $\omega_1, \omega_2, \cdots, \omega_K$, respectively. Customer i in segment k generates the label vector $\boldsymbol{x}_i \sim f_k$, and we observe the labels x_{ij} for all the items $j \in N(i)$, for some preference graph $\mathcal{P}$. For ease of notation, we drop the explicit dependence of models in $\mathcal{F}$ on the parameter ω in the remainder of the discussion. Let $\boldsymbol{x}_i^{\text{obs}} \stackrel{\text{def}}{=} (x_{ij})_{j \in N(i)}$ denote the observed label vector from customer i and define the domain $\mathcal{B}^{(i)} = \{(x_j)_{j \in N(i)} \mid \boldsymbol{x} \in \mathcal{B}\}$. Given any model $f \in \mathcal{F}$, we define $f^{(i)}(\boldsymbol{y}) \stackrel{\text{def}}{=} \sum_{\boldsymbol{x}_i^{\text{mis}}} f(\boldsymbol{y}, \boldsymbol{x}_i^{\text{mis}})$ for each $\boldsymbol{y} \in \mathcal{B}^{(i)}$, where $\boldsymbol{x}_i^{\text{mis}}$ represent the missing labels vector for customer i and the summation is over all feasible missing label vectors when given the observations $\boldsymbol{y}$. Observe that $f^{(i)}$ defines a distribution over $\mathcal{B}^{(i)}$. Finally, let $\left|\boldsymbol{x}_i^{\text{obs}}\right|$ denote the length of the vector $\boldsymbol{x}_i^{\text{obs}}$; we have $\left|\boldsymbol{x}_i^{\text{obs}}\right| = |N(i)|$.

Under the above setup, we adopt the algorithmic framework presented in Algorithm 1 for segmenting the customers. The algorithm takes as inputs the observed customer labels, a model class $\mathcal{F}$ specifying how the labels are generated, and the number of segments, K, and outputs a clustering of the population into K segments. Our framework proceeds in two sequential steps. The first step *embeds* the customers into a low-dimensional Euclidean space, and the second step *clusters* the resulting embeddings to obtain the underlying segments. Our key contribution is the embedding algorithm, and therefore, we focus this section on describing the embedding algorithm. Once the embeddings are obtained, any of the existing

Algorithm 1 Algorithmic Framework for Segmenting Customers

1: **Input:** observed labels $\boldsymbol{x}_1^{\text{obs}}, \boldsymbol{x}_2^{\text{obs}}, \ldots, \boldsymbol{x}_m^{\text{obs}}$; model class $\mathcal{F}$; the number of segments K

2: $(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m) \leftarrow \text{Embed}(\boldsymbol{x}_1^{\text{obs}}, \boldsymbol{x}_2^{\text{obs}}, \ldots, \boldsymbol{x}_m^{\text{obs}}, \mathcal{F})$; $\boldsymbol{v}_i$ is customer i's embedding

3: $(\hat{z}_1, \ldots, \hat{z}_m) \leftarrow \text{Cluster}(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m, K)$; $\hat{z}_i \in [K] \stackrel{\text{def}}{=} \{1, 2, \ldots, K\}$ is customer i's segment label

4: **Output:** Segment labels $(\hat{z}_1, \ldots, \hat{z}_m)$

algorithms can be used to cluster the embeddings—we provide specific recommendations towards the end of the section.

We describe two variants of our embedding algorithm. The first variant assumes that the model class $\mathcal{F}$ is fully-specified—that is, for a given value of $\omega \in \Omega$, the model $f(\cdot; \omega)$ completely specifies how an observation vector $\boldsymbol{x}$ is generated. The second, more general, variant allows the model class to be only partially specified.

1.2.1 Embedding algorithm for fully specified model class

For ease of exposition, we describe the algorithm separately for the cases with equal and unequal customer degrees in the preference graph $\mathcal{P}$. We start with the equal degree case and then deal with the more general unequal degree case.

Equal customer degrees. In this case, all customers have the same number of observations. The algorithm takes the observations in the form of the preference graph $\mathcal{P}$ and the model class $\mathcal{F}$ as inputs and outputs a unidimensional embedding of each customer. It proceeds as follows. Starting with the hypothesis that the population of customers is in fact homogeneous, it looks for evidence of heterogeneity to refute the hypothesis. Under the homogeneity hypothesis, it follows that the m observations $\boldsymbol{x}_1^{\text{obs}}, \boldsymbol{x}_2^{\text{obs}}, \ldots, \boldsymbol{x}_m^{\text{obs}}$ are independent and identically distributed (i.i.d.) samples generated according to a single model in $\mathcal{F}$. Therefore, the algorithm estimates the parameters of a 'pooled' model $f_{\text{pool}} \in \mathcal{F}$ by pooling together all the observations and using a standard technique such as the maximum likelihood estimation (MLE) method. As a concrete example, consider the task of predicting whether a segment of customers likes a movie or not, so that $\mathcal{F}$ is chosen to be the binary logit, or logistic regression, model class where movie j is liked with probability $e^{\omega_j}/(1 + e^{\omega_j})$, independent of the other movies. Then, the parameters of the pooled model can be estimated by solving the following MLE problem:

$$\max_{\omega_1, \omega_2, \ldots, \omega_n} \sum_{i=1}^{m} \sum_{j \in N(i)} \log \left(\frac{e^{\mathbf{1}[x_{ij} = +1] \cdot \omega_j}}{1 + e^{\omega_j}} \right),$$

where $x_{ij} = +1$ if customer i likes movie j and -1 otherwise, and $\mathbf{1}[A]$ is the indicator variable taking value 1 if A is true and 0 otherwise. Because the objective function is separable,

the optimal solution can be shown to be given by $\hat{\omega}_j = \log\left(\frac{\sum_{i:j\in N(i)} \mathbf{1}[x_{ij}=+1]}{\sum_{i:j\in N(i)} \mathbf{1}[x_{ij}=-1]}\right)$ for all items $j \in [n]$.

Once the pooled model is estimated, the algorithm assesses if the hypothesis holds by checking how well the pooled model explains the observed customer labels. Specifically, it quantifies the model fit for customer i by computing the (normalized) negative log-likelihood of observing $\boldsymbol{x}_i^{\text{obs}}$ under the pooled model, i.e., $v_i \stackrel{\text{def}}{=} \frac{1}{d_i} \cdot \left(-\log f_{\text{pool}}^{(i)}(\boldsymbol{x}_i^{\text{obs}})\right)$. A large value of v_i indicates that the observation $\boldsymbol{x}_i^{\text{obs}}$ is not well explained by the pooled model or that customer i's preferences are "far away" from that of the population. The value v_i is a unidimensional representation or embedding of customer i in the Euclidean space. We term it the *model-based embedding score* of the customer because it is obtained by transforming the observations $\boldsymbol{x}_i^{\text{obs}}$ into a real number by means of a model. The entire process is summarized in Algorithm 2.

Algorithm 2 Embedding algorithm with degree normalization

1: **Input:** observed labels $\boldsymbol{x}_1^{\text{obs}}, \boldsymbol{x}_2^{\text{obs}}, \ldots, \boldsymbol{x}_m^{\text{obs}}$ where $\left|\boldsymbol{x}_i^{\text{obs}}\right| = d_i \ \forall\ i$, model class $\mathcal{F}$

2: $f_{\text{pool}} \leftarrow$ estimated pooled model in $\mathcal{F}$

3: For each customer i with observation $\boldsymbol{x}_i^{\text{obs}}$, the embedding $v_i \leftarrow \frac{1}{d_i} \cdot \left(-\log f_{\text{pool}}^{(i)}(\boldsymbol{x}_i^{\text{obs}})\right)$

4: **Output:** embedding scores $\{v_1, v_2, \ldots, v_m\}$

We make the following remarks. First, our embedding algorithm is inspired by the classical statistical technique of analysis-of-variance (ANOVA), which tests the hypothesis of whether a collection of samples are generated from the same underlying population or not. For that, the test starts with the null hypothesis that there is no heterogeneity, fits a single model by pooling together all the data, and then computes the likelihood of the observations under the pooled model. If the likelihood is low (i.e., below a threshold), the test rejects the null hypothesis and concludes that the samples come from different populations. Our algorithm essentially separates customers based on the heterogeneity within these likelihood values.

Second, to understand why our algorithm should be able to separate the segments, consider the following simple case. Suppose a customer from segment k likes any item j with probability $f_k(\text{like}) = \alpha_k$ and dislikes it with probability $f_k(\text{dislike}) = 1 - \alpha_k$ for some $\alpha_k \in [0, 1]$. Different segments differ on the value of the parameter α_k. Suppose q_k denotes the size of segment k, where $\sum_k q_k = 1$ and $q_k > 0$ for all k. Now, when we pool together a large number of observations from these customers, we should essentially observe that the population as a whole likes an item with probability $f_{\text{pool}}(\text{like}) \stackrel{\text{def}}{=} \sum_k q_k \alpha_k$ and dislikes an item with probability $f_{\text{pool}}(\text{dislike}) \stackrel{\text{def}}{=} 1 - f_{\text{pool}}(\text{like})$; this corresponds to the pooled model. Under the pooled model, we obtain the embedding score for customer i as $\frac{1}{|N(i)|} \sum_{j\in N(i)} -\log f_{\text{pool}}(x_{ij})$ where each $x_{ij} \in \{\text{like}, \text{dislike}\}$. Now assuming that $|N(i)|$ is large and because the x_{ij}'s are randomly generated, the embedding score should concentrate around the expectation $\mathbb{E}_{\boldsymbol{X}_i \sim f_k}[-\log f_{\text{pool}}(\boldsymbol{X}_i)]$ where the random variable $\boldsymbol{X}_i$ takes value "like" with probability α_k and "dislike" with probability $1 - \alpha_k$, when customer i belongs to

segment k. The value $\mathbb{E}_{\boldsymbol{X}_i \sim f_k}[-\log f_{\text{pool}}(\boldsymbol{X}_i)]$ is the cross-entropy between the distributions f_k and f_{pool}. Therefore, if the cross-entropies for the different segments are different, our algorithm should be able to separate the segments.[2] We formalize and generalize these arguments in Section 1.3.

Third, our algorithm fits only one model—the 'pooled' model—unlike a classical latent class approach that fits, typically using the expectation-maximization (EM) method, a mixture distribution $g(\boldsymbol{x}) = \sum_k q_k f_k(\boldsymbol{x})$, where all customers in segment k are described by model $f_k \in \mathcal{F}$ and q_k represents the size (or proportion) of segment k. This affords our algorithm two advantages: (a) *speed:* up to 17× faster than the latent class benchmark (see Section 1.4) without the issues of initialization and convergence that are typical of EM-methods; and (b) *flexibility:* allows for fitting models from complex parametric classes $\mathcal{F}$ that more closely explain customer observations.

Unequal customer degrees. We now generalize our embedding algorithm to the case when customers may have unequal degrees in the preference graph $\mathcal{P}$. The issue here is that of normalization—the log-likelihood values $-\log f^{(i)}_{\text{pool}}(\boldsymbol{x}^{\text{obs}}_i)$ depend on the number of observations d_i and should be appropriately normalized in order to be meaningfully compared across customers. It is natural to normalize the log-likelihood values by the corresponding degrees, resulting in Algorithm 2 but applied to the unequal degree setting. Such degree normalization is appropriate when the labels across items are independent, so that the pooled distribution $f_{\text{pool}}(\boldsymbol{x})$ has a product form $f_{\text{pool},1}(x_1) \cdot f_{\text{pool},2}(x_2) \cdots f_{\text{pool},n}(x_n)$. In this case, the log-likelihood under the pooled model becomes $\log f^{(i)}_{\text{pool}}(\boldsymbol{x}^{\text{obs}}_i) = \sum_{j \in N(i)} \log f_{\text{pool},j}(x_{ij})$, which scales in the number of observations d_i.

Degree normalization, however, does not account for dependence structures that may be present in the item labels. For instance, in the extreme case when the observations across all items are perfectly correlated, such that customers either like all items or dislike all items with probability 0.5 each, the log-likelihood value does not depend on the number of observations. Yet, degree normalization divides the value by the degree, unfairly penalizing customers with only a few observations. To address this issue, we use entropy normalization:

$$v_i = \frac{-\log f^{(i)}_{\text{pool}}(\boldsymbol{x}^{\text{obs}}_i)}{H(f^{(i)}_{\text{pool}})} = \frac{-\log f^{(i)}_{\text{pool}}(\boldsymbol{x}^{\text{obs}}_i)}{-\sum_{\boldsymbol{y} \in \mathcal{B}^{(i)}} f^{(i)}_{\text{pool}}(\boldsymbol{y}) \log f^{(i)}_{\text{pool}}(\boldsymbol{y})} \tag{1.1}$$

where $H(f^{(i)}_{\text{pool}})$ denotes the *entropy* of distribution $f^{(i)}_{\text{pool}}$. When the labels across items are i.i.d., it can be seen that entropy normalization reduces to degree-normalization, upto constant factors. In addition, if the population has homogeneous preferences, all the customer embeddings concentrate around the value 1 (see Section 1.3.1). Therefore, deviations of embeddings from 1 indicates heterogeneity in customer preferences, allowing us to separate the different segments.

[2]Note that the cross-entropy is **not** a distance measure between distributions unlike the standard KL (Kullback-Leibler) divergence. Consequently, even when $f_k = f_{\text{pool}}$ for some segment k, the cross-entropy is not zero. Our algorithm relies on the cross-entropies being distinct to recover the underlying segments.

Entropy normalizations have been commonly used in the literature for normalizing mutual information [SG02]—our normalization is inspired by that. In addition to accounting for dependency structures within the pooled distribution, it has the effect of weighting each observation by the strength of the evidence it provides. Because the log-likelihood value $-\log f^{(i)}_{\text{pool}}(\boldsymbol{x}^{\text{obs}}_i)$ only provides incomplete evidence of how well f_{pool} captures the preferences of customer i when there are missing observations, we assess the confidence in the evidence by dividing the log-likelihood value by the corresponding entropy $H(f^{(i)}_{\text{pool}})$ of the distribution $f^{(i)}_{\text{pool}}$. Higher values of entropy imply lower confidence. Therefore, when entropy is high, the embedding score is low, indicating that there is insufficient evidence to conclude that customer i's observations are not well-explained by f_{pool}. Algorithm 3 summarizes our embedding algorithm with entropy normalization.

Algorithm 3 Embedding algorithm with entropy normalization

same as Algorithm 2 except replace step 3 with:

Compute the embedding v_i of customer i via equation (1.1)

Entropy may be difficult to compute because, in general, it requires summing over an exponentially large space. For such cases, either the entropy computation may be approximated using existing techniques for (approximate) inference in probabilistic graphical models [WJ08] or degree normalization of Algorithm 2 may be used as an approximation.

1.2.2 Embedding algorithm for partially specified model class

The discussion so far assumed that the model class $\mathcal{F}$ is fully-specified with $f(\cdot;\omega)$ specifying the complete joint distribution of each observation $\boldsymbol{x}$. In many practical settings, specifying the complete joint distribution structure is difficult especially when the item universe is large (for instance, millions in our eBay case study) and there are complex cross-effects among items, such as the correlation between the rating and purchase signal for the same item or complementarity effects among products from related categories (clothing, shoes, accessories, etc.). To handle such situations, we extend our embedding algorithm to the case when the model is only partially specified.

The precise setting we consider is as follows. The universe $[n]$ of items is partitioned into $B > 1$ "categories" $\{\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_B\}$ such that $\mathcal{I}_b$ is the set of items in category $b \in [B] \stackrel{\text{def}}{=} \{1, 2, \ldots, B\}$. A model describing the observations within each category is specified, but any interactions across categories are left unspecified. We let $\mathcal{F}_b(\Omega_b) = \{f(\boldsymbol{x}_b;\omega) : \omega \in \Omega_b\}$ denote the model class for category b, so that segment k is characterized by the B models $(f_{k1}, f_{k2}, \ldots, f_{kB})$ with $f_{kb} \in \mathcal{F}_b$ for all $1 \le b \le B$. Further, $\boldsymbol{x}^{\text{obs}}_{ib}$ denotes the vector of observations of customer i for items in category b; if there are no observations, we set $\boldsymbol{x}^{\text{obs}}_{ib} = \phi$.

Under this setup, we run our embedding algorithm (Algorithm 2 or 3) separately for each category of items. This results in a B-dimensional vector $\boldsymbol{v}_i = (v_{i1}, v_{i2}, \ldots, v_{iB})$ for each

customer i, where v_{ib} is the embedding score computed by our algorithm for customer i and category b. When $\boldsymbol{x}_{ib}^{\text{obs}} = \phi$, we set $v_{ib} = \phi$. These vectors are compactly represented as the following $m \times B$ matrix with row i corresponding to customer i:

$$\mathcal{V} = \begin{bmatrix} v_{11} & v_{12} & \dots & v_{1B} \\ v_{21} & v_{22} & \dots & v_{2B} \\ \vdots & \vdots & \ddots & \vdots \\ v_{m1} & v_{m2} & \dots & v_{mB} \end{bmatrix}$$

When matrix $\mathcal{V}$ is complete, the algorithm stops and outputs $\mathcal{V}$ with $\boldsymbol{v}_i$ representing the embedding vector for customer i. Instead, if there are missing entries, we obtain the embeddings for the customers using low-rank matrix decomposition (or factorization) techniques, similar to those adopted in collaborative filtering applications [KBV+09]. These techniques assume that the matrix $\mathcal{V}$ with missing entries can be factorized into a product of two low-rank matrices—one specifying the customer representation and the other the item category representation, in a low-dimensional space. The low-rank structure naturally arises from assuming that only a small number of (latent) factors influence the cross-effects across categories. With this assumption, we compute an r-dimensional representation $\boldsymbol{u}_i \in \mathbb{R}^r$ for each customer i and $\boldsymbol{y}_b \in \mathbb{R}^r$ for each item category b by solving the following optimization problem:

$$\min_{\mathcal{U},\mathcal{Y}} \sum_{i=1}^{m} \sum_{b=1}^{B} \mathbf{1}[v_{ib} \neq \phi] \cdot \left(v_{ib} - \boldsymbol{u}_i^\top \boldsymbol{y}_b\right)^2 \tag{1.2}$$

where row i of matrix $\mathcal{U} \in \mathbb{R}^{m \times r}$ is $\boldsymbol{u}_i^\top$ and row b of matrix $\mathcal{Y} \in \mathbb{R}^{B \times r}$ is $\boldsymbol{y}_b^\top$. Note that the rank $r \ll \min(m, B)$. The embedding vector for customer i is then given by $\boldsymbol{u}_i^\top \mathcal{Y}^\top$. Algorithm 4 summarizes the above process.

Algorithm 4 Embedding algorithm for a partially specified model class $\mathcal{F}$

1: **Input:** observed labels $\boldsymbol{x}_1^{\text{obs}}, \boldsymbol{x}_2^{\text{obs}}, \dots, \boldsymbol{x}_m^{\text{obs}}$, item partitioning $\{\mathcal{I}_1, \dots, \mathcal{I}_B\}$, model class $\mathcal{F}_b$ for each category $b \in [B]$, the rank $r \ll \min(m, B)$ of low-rank decomposition

2: $f_{\text{pool},b} \leftarrow$ estimated pooled model in $\mathcal{F}_b$ for all $1 \leq b \leq B$

3: Compute v_{ib} for each customer i and category b using Algorithm 2 or 3 whenever $\boldsymbol{x}_{ib}^{\text{obs}} \neq \phi$, otherwise set $v_{ib} = \phi$

4: Create the $m \times B$ matrix $\mathcal{V}$ with row i given by the vector $\boldsymbol{v}_i = (v_{i1}, v_{i2}, \dots, v_{iB})$

5: If $\mathcal{V}$ is incomplete, compute rank r-factorization $\mathcal{V} \approx \mathcal{U}\mathcal{Y}^\top$ where $\mathcal{U} \in \mathbb{R}^{m \times r}, \mathcal{Y} \in \mathbb{R}^{B \times r}$ by solving optimization problem (1.2)

6: **Output:** $\mathcal{V}$ if it is complete and $\mathcal{U}\mathcal{Y}^\top$ otherwise

We conclude this subsection with a discussion of some practical issues regarding the implementation of Algorithm 4. First, the algorithm assumes that the item categories are specified as input—they could correspond to a well-defined partitioning of the items, such as movie genres, product categories, etc., but this is not necessary and the categories can be customized based on the particular application context. See Section 1.3.2 for some examples. As an extreme, each item could belong to its own category, in which case the algorithm reduces to the standard matrix factorization technique (when the matrix $\mathcal{V}$ is incomplete). Second, in most real-world settings, $\mathcal{V}$ will be incomplete, so that the low-rank decomposition step will be executed. Since our focus is on segmenting large customer populations, m will often be large (ranging from thousands to even millions, as in our eBay case study in Section 1.6), and the number of categories B can also be large—for instance, eBay has hundreds of product categories. Consequently, we require scalable techniques to compute the low-rank factorization, for which we can tap into the machine learning literature that has proposed numerous algorithms in the past decade or so; see for instance [MS07, TPNT09, MHT10]. Third, a standard way of choosing the rank r is via cross-validation, but there is also some recent work that can automatically determine the "right" rank [BLMK12, TF13]. Finally, as discussed in Section 1.1, an individual customer might interact with only a tiny fraction of the item universe, as a result of which the matrix $\mathcal{V}$ will contain a large number of missing entries, significantly more than in Netflix-like rating systems (refer to Section 1.6 for some concrete numbers). Consequently, traditional matrix factorization techniques can be prone to overfitting, and the literature has proposed bayesian methods [SM08, IR10] that automatically control the model capacity and result in improved prediction performance.

1.2.3 Clustering the embeddings to obtain segments

The final step is to cluster the customer embeddings to obtain the different segments. This step requires the choice of a clustering algorithm and the number of segments, K.

Clustering algorithm. Since the embeddings are vectors in the Euclidean space, any of the existing techniques designed for clustering real-valued vectors may be used. Popular candidates include k-means, mean-shift, kernel k-means [DGK04], spectral clustering and spectral projection techniques. The choice of the technique depends on the specific application context and the corresponding strengths and weaknesses of the different clustering techniques. The k-means algorithm is the most popular candidate. It iteratively chooses a set of centroids and partitions the data points by assigning each point to its closest centroid. It is widely used in practice and can scale to large numbers of data points. When the embedding is unidimensional, the mean-shift algorithm [CM02] may be used. It can identify clusters of arbitrary shape by means of an appropriately chosen kernel and also automatically determines the number of clusters. Spectral clustering [NJW01] and spectral projection techniques [AM05, KSV05] are preferred candidates when the embeddings are "high" dimensional because they first project the vectors to a low-dimensional space before clustering. Another approach to deal with high-dimensional embeddings is to employ variable selection methods [GKT95, LFJ04, RD06], such as imposing automatic relevance determination (ARD) priors [RL04, YH11] that automatically

select the subset of features that are most relevant for clustering the data points. Finally, hierarchical clustering [Jai10] algorithms may be used if a particular application calls for nested clusters.

Number of segments. Any of the numerous techniques proposed in the literature including cross-validation [Wan10] and information-theoretic measures [MP00] like the AIC, BIC, etc. can be used to pick the appropriate number of clusters K. In addition, nonparametric bayesian methods like the dirichlet process mixture model [Ras00, GR10, Teh11] can automatically infer the number of clusters from the data, and provide the ability to adapt the number of active segments as more data (for instance, from new customers) is fed into the model over time. When the embeddings are unidimensional, our technique also provides data-driven guidance on choosing a set of possible numbers of segments. Our theoretical analysis in Section 1.3 shows that unidimensional embeddings of customers in the same segment concentrate around the same value, whereas those of customers in different segments concentrate around distinct values, provided we have sufficient number of observations from each customer. Consequently, if we estimate the empirical distribution of the embeddings, using a general purpose technique like kernel density estimation (KDE), then the number of modes of the distribution should correspond to the underlying number of clusters.[3] Since the modes are sometimes difficult to distinguish, this approach provides us with a set of candidate numbers of segments. The precise number K may then be picked through cross-validation. But this approach does not scale to higher-dimensional embeddings.

1.3 Theoretical Results

Our segmentation algorithm is analytically tractable and in this section, we derive theoretical conditions for how "separated" the underlying segments must be to guarantee asymptotic recovery using our algorithm. Our results are similar in spirit to existing theoretical results for clustering observations from mixture models, such as mixture of multivariate Gaussians [AM05, KSV05].

For the purposes of the theoretical analysis, we focus on the following standard setting—there is a population of m customers comprising K distinct segments such that a proportion $q_k > 0$ of the population belongs to segment k, for each $k \in [K]$; we have that $\sum_{k\in[K]} q_k = 1$. Segment k is described by distribution $\pi_k \colon \{-1,+1\}^n \to [0,1]$ over the domain $\mathcal{B} := \{-1,+1\}^n$. Note that this corresponds to the scenario when $\mathcal{X}_j = \{-1,+1\}$ for all items j (see the notation in Section 1.2). We sometimes refer to $+1$ as *like* and -1 as *dislike* in the remainder of the section. Customer i's latent segment is denoted by $z_i \in [K]$, so that if $z_i = k$, then i samples a vector $\boldsymbol{x}_i \in \mathcal{B}$ according to distribution π_k, and then assigns the label x_{ij} for item j. We focus on asymptotic recovery of the true segment labels $\boldsymbol{z} = (z_1, z_2, \ldots, z_m)$, as the number of items $n \to \infty$.

The performance of our algorithm depends on the separation among the hyper-parameters

[3] This is also the idea behind the mean-shift clustering algorithm which clusters data points by assigning them to the nearest mode of the KDE.

describing the segment distributions π_k, as well as the number of data points available per customer. Therefore, we assume that the segment distributions are "well-separated" (the precise technical conditions are described below) and the number of data points per customer goes to infinity as $n \to \infty$. The proof of all results are given in Appendix A.1.

1.3.1 Fully specified model class: independent item preferences

We first consider the case where π_k belongs to a fully specified model class $\mathcal{F}(\Omega)$, and customer labels across items are independent. More precisely, we consider the following generative model:

Definition 1.1 (latent class independent (LC-IND) model)**.** Each segment k is described by distribution $\pi_k \colon \{-1,+1\}^n \to [0,1]$, which is such that item labels $\{x_j\}_{j\in[n]}$ are independent and identically distributed. Let $\alpha_k = \mathbb{P}_{\boldsymbol{x}\sim\pi_k}[x_j = +1]$ for all items $j \in [n]$, i.e. α_k is the probability that a customer from segment k likes an item. Customer i in segment k samples vector $\tilde{\boldsymbol{x}}_i$ according to distribution π_k and provides label $\tilde{x}_{ij}$ for item j.

We assume that the segment parameters are bounded away from 0 and 1, i.e. there exists a constant $\alpha_{\min} > 0$ such that $0 < \alpha_{\min} \leq \alpha_k \leq 1 - \alpha_{\min} < 1$ for all segments $k \in [K]$. Further, let $H(\beta_1,\beta_2) = -\beta_1 \log \beta_2 - (1-\beta_1)\log(1-\beta_2)$ denote the *cross-entropy* between the Bernoulli distributions $\mathrm{Ber}(\beta_1)$ and $\mathrm{Ber}(\beta_2)$, with $0 \leq \beta_1, \beta_2 \leq 1$, and $H(\alpha) = -\alpha\log(\alpha) - (1-\alpha)\log(1-\alpha)$ denote the binary entropy function, where $0 \leq \alpha \leq 1$. Let $\boldsymbol{V}_i$ denote the unidimensional embedding score computed by Algorithm 3, note that it is a random variable under the generative model above.

Given the above, we derive necessary and sufficient conditions to guarantee (asymptotic) recovery of the true customer segments under the LC-IND model.

Necessary conditions for recovery of true segments. We first present an important result concerning the concentration of the customer embeddings computed by our algorithm.

Lemma 1.1 (Concentration of embedding scores under LC-IND model)**.** *Suppose that the preference graph $\mathcal{P}$ is ℓ-regular, i.e. $d_i = \ell$ for all customers $1 \leq i \leq m$. Define the quantity $\alpha_{\mathrm{pool}} \stackrel{\mathrm{def}}{=} \sum_{k=1}^K q_k\alpha_k$. Then given any $0 < \varepsilon < 1$, the embedding scores computed by Algorithm 3 are such that:*

$$\mathbb{P}\left[\left|\boldsymbol{V}_i - \frac{H(\alpha_{z_i},\alpha_{\mathrm{pool}})}{H(\alpha_{\mathrm{pool}})}\right| > \varepsilon\frac{H(\alpha_{z_i},\alpha_{\mathrm{pool}})}{H(\alpha_{\mathrm{pool}})}\right] \leq$$
$$4\exp\left(\frac{-2\ell\varepsilon^2\alpha_{\min}^2}{81}\right) + 12\exp\left(\frac{-2m\cdot\ell\cdot\varepsilon^2\bar{\alpha}_{\mathrm{pool}}^2\log^2(1-\bar{\alpha}_{\mathrm{pool}})}{81\cdot\left(1-\log(1-\bar{\alpha}_{\mathrm{pool}})\right)^2}\right)$$

where $\bar{\alpha}_{\mathrm{pool}} \stackrel{\mathrm{def}}{=} \min(\alpha_{\mathrm{pool}}, 1-\alpha_{\mathrm{pool}})$. In other words, the embedding scores of customers in segment k concentrate around the ratio $\frac{H(\alpha_k,\alpha_{\mathrm{pool}})}{H(\alpha_{\mathrm{pool}})}$, with high probability as the number of observations from each customer $\ell \to \infty$.

Lemma 1.1 reveals the necessary conditions our algorithm requires to recover the true customer segments. To understand the result, first suppose that $\alpha_{\text{pool}} \neq (1/2)$. Then, the above result states that the model-based embedding scores of customers in segment k concentrate around $\frac{H(\alpha_k, \alpha_{\text{pool}})}{H(\alpha_{\text{pool}})}$ which is proportional to $-\alpha_k \log \frac{\alpha_{\text{pool}}}{1-\alpha_{\text{pool}}} - \log(1-\alpha_{\text{pool}})$. Consequently, we require that $\alpha_k \neq \alpha_{k'}$ whenever $k \neq k'$ to ensure that the embedding scores of customers in different segments concentrate around distinct values. The result also states that the embedding scores of customers with similar preferences (i.e., belonging to the same segment) are close to each other, i.e. concentrate around the same quantity, whereas the scores of customers with dissimilar preferences (i.e. belonging to different segments) are distinct from each other. For this reason, although it is not a priori clear, our segmentation algorithm is consistent with the classical notion of distance- or similarity-based clustering, which attempts to maximize intra-cluster similarity and inter-cluster dissimilarity. When $\alpha_{\text{pool}} = (1/2)$, it follows that $H(\alpha_k, \alpha_{\text{pool}}) = H(\alpha_{\text{pool}})$ for *any* $0 \leq \alpha_k \leq 1$, and therefore all the customer embedding scores concentrate around 1. In this scenario, our algorithm cannot separate the customers even when the parameters α_k of different segments are distinct. Note that $\alpha_{\text{pool}} = \sum_k q_k \alpha_k$, which is the probability that a random customer from the population likes an item. Therefore, when $\alpha_{\text{pool}} = (1/2)$, the population is indifferent in its preferences for each item, resulting in all the customers being equidistant from the pooled customer.

The above discussion leads to the following theorem:

Theorem 1.2 (Necessary conditions for recovery of true segments under LC-IND model)**.** *Under the setup of Lemma 1.1, the following conditions are necessary for recovery of the true customer segments:*

1. *All segment parameters are distinct, i.e.* $\alpha_k \neq \alpha_{k'}$ *whenever* $k \neq k'$*, and* 2. $\alpha_{\text{pool}} \neq \frac{1}{2}$.

It is easy to see that the first condition is necessary for *any* segmentation algorithm. We argue that the second condition, i.e. $\alpha_{\text{pool}} \neq \frac{1}{2}$, is also necessary for the standard latent class (LC) segmentation technique. Specifically, consider two segments such that $q_1 = q_2 = 0.5$ and let $\alpha_1 = 1, \alpha_2 = 0$. Then, it follows that $\alpha_{\text{pool}} = q_1\alpha_1 + q_2\alpha_2 = \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 0 = \frac{1}{2}$. Let us consider only a single item, i.e. $n = 1$. Then, under this parameter setting, all customers in segment 1 will give the label $+1$ and all customers in segment 2 will give label -1. Recall that the LC method estimates the model parameters by maximizing the log-likelihood of the observed labels, which in this case looks like:

$$\log \mathcal{L} = \frac{m}{2} \log \left(q_1\alpha_1 + q_2\alpha_2\right) + \frac{m}{2} \log \left(q_1 \cdot (1-\alpha_1) + q_2 \cdot (1-\alpha_2)\right)$$

Then it can be seen that the solution $\hat{q}_1 = \hat{q}_2 = 0.5$ and $\hat{\alpha}_1 = \hat{\alpha}_2 = 0.5$ achieves the optimal value of the above log-likelihood function, and therefore is a possible outcome recovered by the LC method. This shows that the condition $\alpha_{\text{pool}} \neq \frac{1}{2}$ is also necessary for the standard LC method.

We also note that our results can be extended to the case when $\mathcal{P}$ is not ℓ-regular but with additional notation.

Sufficient conditions for recovery of true segments. Having established the necessary conditions, we now analyze the asymptotic *misclassification rate*, defined as the expected

fraction of customers incorrectly classified, of our algorithm. In particular, we consider the following nearest-neighbor (NN) classifier $\hat{\boldsymbol{I}}(\cdot)$, where customer i is classified as:

$$\hat{\boldsymbol{I}}(i) = \underset{k=1,2,\ldots,K}{\arg\min} \frac{|\boldsymbol{V}_i - H_k|}{H_k}$$

where $H_k \stackrel{\text{def}}{=} \frac{H(\alpha_k, \alpha_{\text{pool}})}{H(\alpha_{\text{pool}})}$. Given the necessary conditions established above and to ensure that we can uniquely identify the different segments, we assume that the segments are indexed such that $\alpha_1 < \alpha_2 < \ldots < \alpha_K$. Then, we can establish the following recoverability result:

Theorem 1.3 (Asymptotic recovery of true segments under LC-IND model)**.** *Under the setup of Lemma 1.1, suppose* $0 < \alpha_{\min} \leq \alpha_1 < \alpha_2 < \cdots < \alpha_K \leq 1 - \alpha_{\min}$ *and* $\alpha_{\text{pool}} \neq \frac{1}{2}$*. Further, denote* $\lambda = \min_{k=1,2,\ldots,K-1}(\alpha_{k+1} - \alpha_k)$*. Given any* $0 < \delta < 1$*, suppose that*

$$\ell \geq \frac{648}{\lambda^2} \cdot \left(\frac{\log \alpha_{\min}}{\log(1 - \alpha_{\min}) \cdot \alpha_{\min}} \right)^2 \cdot \frac{1}{\log^2 \frac{\alpha_{\text{pool}}}{1-\alpha_{\text{pool}}}} \cdot \log(16/\delta)$$

Then, it follows that

$$\frac{1}{m} \sum_{i=1}^{m} \mathbb{P}\left[\hat{\boldsymbol{I}}(i) \neq z_i \right] < \delta$$

Further, when $\ell = \log n$ *and* $m \geq \left(\frac{1 - \log(1 - \bar{\alpha}_{\text{pool}})}{\log(1 - \bar{\alpha}_{\text{pool}})} \right)^2$*, we have:*

$$\frac{1}{m} \sum_{i=1}^{m} \mathbb{P}\left[\hat{\boldsymbol{I}}(i) \neq z_i \right] = O\left(n^{-\frac{2\Lambda^2 \alpha_{\min}^2}{81}} \right) \quad \text{where the constant } \Lambda \stackrel{\text{def}}{=} \frac{\lambda}{2} \cdot \left(\frac{\left| \log \frac{\alpha_{\text{pool}}}{1-\alpha_{\text{pool}}} \right|}{|\log \alpha_{\min}|} \right)$$

Theorem 1.3 provides an upper bound on the misclassification rate of our segmentation algorithm in recovering the true customer segments. The first observation is that as the number of labels from each customer $\ell \to \infty$, the misclassification rate of the NN classifier goes to zero. The result also allows us determine the number of samples ℓ needed per customer to guarantee an error rate δ. In particular, ℓ depends on three quantities:

1. $\frac{1}{\lambda^2}$ where λ is the minimum separation between the segment parameters. This is intuitive—the "closer" the segments are to each other (i.e. smaller value of λ), the more samples are required per customer to successfully identify the true segments.

2. $\frac{1}{\log^2 \frac{\alpha_{\text{pool}}}{1-\alpha_{\text{pool}}}}$ where recall that α_{pool} is the probability that a random customer from the population likes an item. If $\alpha_{\text{pool}} \approx \frac{1}{2}$, then $\log^2 \frac{\alpha_{\text{pool}}}{1-\alpha_{\text{pool}}} \approx 0$ so that we require a large number of samples per customer. As α_{pool} deviates from $\frac{1}{2}$, the quantity $\log^2 \frac{\alpha_{\text{pool}}}{1-\alpha_{\text{pool}}}$ increases, so fewer samples are sufficient. This also makes sense—when $\alpha_{\text{pool}} = (1/2)$, our algorithm cannot identify the underlying segments, and the farther α_{pool} is from $(1/2)$, the easier it is to recover the true segments.

3. $\alpha_{\min}$, where as $\alpha_{\min} \to 0$, the number of samples required diverges. Note that $\alpha_{\min}$ (resp. $1 - \alpha_{\min}$) specifies a lower (upper) bound on the segment parameters α_k—a small value of $\alpha_{\min}$ indicates that there exists segments with values of α_k close to either 0 or 1; and since the number of samples required to reliably estimate α_k (resp. $1 - \alpha_k$) grows as $\frac{1}{\alpha_k^2}$ (resp. $\frac{1}{(1-\alpha_k)^2}$), ℓ must diverge as $\alpha_{\min} \to 0$.

Our result shows that as long as each customer provides at least $\log n$ labels, the misclassification rate goes to zero, i.e. we can accurately recover the true segments with high probability, as the number of items $n \to \infty$. Although the number of labels required from each customer must go to infinity, it must only grow logarithmically in the number of items n. Further, this holds for *any* population size "large enough".

Note that the NN classifier above assumes access to the "true" normalized cross-entropies H_k. In practice, we use "empirical" NN classifiers, which replace H_k by the corresponding cluster centroids of the embedding scores. Lemma 1.1 guarantees the correctness of this approach under appropriate assumptions, because the embedding scores of segment k customers concentrate around H_k.

1.3.2 Partially specified model class: independent within-category item preferences

We can extend the results derived above to the case when the distributions π_k belong to a partially specified model class, as discussed in Section 1.2.2. Specifically, suppose that the item set $[n]$ is partitioned into $B > 1$ (disjoint) categories: $\mathcal{I}_1 \uplus \mathcal{I}_2 \cdots \uplus \mathcal{I}_B$. The preferences of customers vary across the different categories, specifically we consider the following generative model:

Definition 1.2 (latent class independent category (LC-IND-CAT) model)**.** Each segment k is described by distribution $\pi_k \colon \{-1, +1\}^n \to [0, 1]$, which is such that labels $\{x_{j_b}\}_{j_b \in \mathcal{I}_b}$ for items within a single category $b \in [B]$ are independent and identically distributed; but labels for items in different categories can have arbitrary correlations. Let $\boldsymbol{\alpha}_k = (\alpha_{k1}, \alpha_{k2}, \cdots, \alpha_{kB})$ be such that $\mathbb{P}_{\boldsymbol{x} \sim \pi_k}[x_{j_b} = +1] = \alpha_{kb}$ for each item $j_b \in \mathcal{I}_b$ and each category $b \in [B]$. Customer i in segment k samples vector $\tilde{\boldsymbol{x}}_i$ according to distribution π_k and provides label $\tilde{x}_{ij}$ for each item j.

The above model is general and can be used to account for correlated item preferences (as opposed to independent preferences considered in Section 1.3.1). As a specific example, suppose that for each item, we have two customer observations available: whether the item was purchased or not, and a like/dislike rating (note that one of these can be missing). Clearly these two observations are correlated and we can capture this scenario in the LC-IND-CAT model as follows: there are two item "categories"—one representing the purchases and the other representing the ratings. In other words, we create two copies of each item and place one copy in each category. Then, we can specify a joint model over the item copies such that purchase decisions for different items are independent, like/dislike ratings for different

items are also independent but the purchase decision and like/dislike rating for the *same* item are dependent on each other. Similar transformations can be performed if we have more observations per item or preferences are correlated for a group of items. Therefore, the above generative model is fairly broad and captures a wide variety of customer preference structures.

As done for the LC-IND model, we assume that the underlying segment parameters are bounded away from 0 and 1, i.e. there exists constant $\alpha_{\min} > 0$ such that $0 < \alpha_{\min} \leq \alpha_{kb} \leq 1 - \alpha_{\min} < 1$ for all segments $k \in [K]$, and all item categories $b \in [B]$. Let d_{ib} be the number of observations for customer i in category b and let $\overrightarrow{\boldsymbol{V}}_i$ denote the embedding vector computed by Algorithm 4 for customer i, note that it is a B-dimensional random vector under the generative model above.

Necessary conditions for recovery of true segments. We first state an analogous concentration result for the customer embedding vectors computed by our algorithm.

Lemma 1.4 (Concentration of embedding vectors under LC-IND-CAT model)**.** *Suppose that the preference graph $\mathcal{P}$ is such that each customer labels exactly $\ell_b > 0$ items in category b, i.e. $d_{ib} = \ell_b$ for all $1 \leq i \leq m$. Define the quantities $\alpha_{b,\text{pool}} \stackrel{\text{def}}{=} \sum_{k=1}^{K} q_k \alpha_{kb}$ for each item category b, $\ell_{\min} \stackrel{\text{def}}{=} \min_{b \in [B]} \ell_b$, and $\hat{\alpha}_{\text{pool}} \stackrel{\text{def}}{=} \min_{b \in [B]} \bar{\alpha}_{b,\text{pool}}$ where $\bar{\alpha}_{b,\text{pool}} \stackrel{\text{def}}{=} \min\left(\alpha_{b,\text{pool}}, 1 - \alpha_{b,\text{pool}}\right)$. Then given any $0 < \varepsilon < 1$, the embedding vectors computed by Algorithm 4 are such that:*

$$\mathbb{P}\left[\|\overrightarrow{\boldsymbol{V}}_i - \boldsymbol{H}_{z_i}\|_1 > \varepsilon \|\boldsymbol{H}_{z_i}\|_1\right] \leq$$
$$4B \cdot \exp\left(\frac{-2\ell_{\min}\varepsilon^2\alpha_{\min}^2}{81}\right) + 12B \cdot \exp\left(\frac{-2m \cdot \ell_{\min} \cdot \varepsilon^2 \hat{\alpha}_{\text{pool}}^2 \log^2(1 - \hat{\alpha}_{\text{pool}})}{81\big(1 - \log(1 - \hat{\alpha}_{\text{pool}})\big)^2}\right)$$

In the result above, $\boldsymbol{H}_k = (H_{k1}, H_{k2}, \cdots, H_{kB})$ is a B-dimensional vector such that $H_{kb} = \frac{H(\alpha_{kb}, \alpha_{b,\text{pool}})}{H(\alpha_{b,\text{pool}})}$ (recall notation from Section 1.3.1) and $\|\cdot\|_1$ denotes the L_1-norm. Lemma 1.4 implies the following necessary conditions:

Theorem 1.5 (Necessary conditions for recovery under LC-IND-CAT model)**.** *Under the setup of Lemma 1.4, the following conditions are necessary for recovery of the true customer segments:*

1. *$\alpha_{b,\text{pool}} \neq \frac{1}{2}$ for some category $b \in [B]$.*
2. *Let $B' = \{b \in [B] : \alpha_{b,\text{pool}} \neq \frac{1}{2}\}$ and let $(\alpha_{kb})_{b \in B'}$ be the subvector of $\boldsymbol{\alpha}_k$ consisting of components corresponding to item categories B'. Then $(\alpha_{kb})_{b \in B'} \neq (\alpha_{k'b})_{b \in B'}$ whenever $k \neq k'$.*

Similar to the LC-IND model case, $\alpha_{b,\text{pool}} = (1/2)$ for *all* item categories implies that the population is indifferent in its preferences for each item in each category. However, we require the population to have well-defined preferences for at least one category in order to be able

to separate the segments. Further, since $H_{kb} \propto -\alpha_{kb} \log \frac{\alpha_{b,\text{pool}}}{1-\alpha_{b,\text{pool}}} - \log(1-\alpha_{b,\text{pool}})$, we need $\alpha_{kb} \neq \alpha_{k'b}$ for at least one item category b where $\alpha_{b,\text{pool}} \neq \frac{1}{2}$ to ensure that the vectors $\boldsymbol{H}_k$ and $\boldsymbol{H}_{k'}$ are distinct, for any two segments $k \neq k'$.

Sufficient conditions for recovery of true segments. As for the case of the LC-IND model, we consider another NN classifier, $\hat{\boldsymbol{I}}_2(\cdot)$, to evaluate the asymptotic misclassification rate of our segmentation algorithm, where customer i is classified as:

$$\hat{\boldsymbol{I}}_2(i) = \underset{k=1,2,\ldots,K}{\arg\min} \frac{\|\overrightarrow{\boldsymbol{V}_i} - \boldsymbol{H}_k\|_1}{\|\boldsymbol{H}_k\|_1}$$

Given the above necessary conditions, we can establish the following recoverability result:

Theorem 1.6 (Asymptotic recovery of true segments under LC-IND-CAT model)**.** *Suppose that the conditions in Theorem 1.5 are satisfied. Denote* $\boldsymbol{w} = (w_1, w_2, \cdots, w_B)$ *with* $w_b = \left|\log \frac{\alpha_{b,\text{pool}}}{1-\alpha_{b,\text{pool}}}\right|$ *and* $\gamma = \min_{k \neq k'} \|\boldsymbol{w} \odot (\boldsymbol{\alpha}_k - \boldsymbol{\alpha}_{k'})\|_1$ *where* $\odot$ *represents element-wise product. Under the setup of Lemma 2 and given any* $0 < \delta < 1$*, suppose that*

$$\ell_{\min} \geq \frac{648B^2}{\gamma^2} \cdot \left(\frac{\log \alpha_{\min}}{\log^2(1-\alpha_{\min}) \cdot \alpha_{\min}}\right)^2 \log(16B/\delta)$$

Then, it follows that

$$\frac{1}{m} \sum_{i=1}^{m} \mathbb{P}\left[\hat{\boldsymbol{I}}_2(i) \neq z_i\right] < \delta$$

Further, when $\ell_{\min} = \log n$ *and* $m \geq \left(\frac{1-\log(1-\hat{\alpha}_{\text{pool}})}{\log(1-\hat{\alpha}_{\text{pool}})}\right)^2$*, for fixed* B *we have:*

$$\frac{1}{m} \sum_{i=1}^{m} \mathbb{P}\left[\hat{\boldsymbol{I}}_2(i) \neq z_i\right] = O\left(n^{\frac{-2\Gamma^2\alpha_{\min}^2}{81}}\right) \quad \textit{where the constant } \Gamma \stackrel{\text{def}}{=} \frac{\gamma}{2B} \cdot \left|\frac{\log(1-\alpha_{\min})}{\log \alpha_{\min}}\right|$$

We make a few remarks about Theorem 1.6. First, as $\ell_{\min} \to \infty$, i.e. the number of labels in each item category $\ell_b \to \infty$, the misclassification rate of the NN classifier goes to zero. Second, to achieve misclassification rate of at most δ, the number of samples $\ell_{\min}$ scales as

1. $\frac{1}{\gamma^2}$ where γ is the minimum weighted L_1-norm of the difference between the parameter vectors of any two segments. This is similar to a standard "separation condition"—the underlying segment vectors $\boldsymbol{\alpha}_k$ should be sufficiently distinct from each other, as measured by the L_1-norm. However, instead of the standard L_1-norm, we require a weighted form of the norm, where the weight of each component is given by $w_b = \left|\log \frac{\alpha_{b,\text{pool}}}{1-\alpha_{b,\text{pool}}}\right|$. If $\alpha_{b,\text{pool}} \approx \frac{1}{2}$, then $w_b \approx 0$ so that the separation in dimension b is weighed lower than categories where $\alpha_{b,\text{pool}}$ is sufficiently distinct from $\frac{1}{2}$. This follows from the necessary condition in Theorem 1.5 and is a consequence of the simplicity of our algorithm that relies on measuring deviations of customers from the population preference.

2. B^2, this is expected—as the number of categories increases, we require more samples to achieve concentration in *all* dimensions of the embedding vector $\overrightarrow{\boldsymbol{V}}_i$.

3. $\alpha_{\min}$, the dependence on which is similar to the LC-IND model case, but with an extra factor of $\log^2(1-\alpha_{\min})$ in the denominator, indicating a more stronger dependence on $\alpha_{\min}$.

Finally, it follows that a logarithmic number of labels in *each* category is sufficient to guarantee recovery of the true segments with high probability as the total number of items $n \to \infty$, provided the population size m is "large enough".

1.4 Computational study: Accuracy of model-based embedding technique

The theoretical analysis above provides asymptotic recovery guarantees for our approach when all customers have equal degrees in the preference graph $\mathcal{P}$ and customer degrees grow to infinity. This section supplements our theoretical results with the results from a computational study that tests the performance of our technique when customer degrees are unequal and finite. The study analyzes the misclassification rate of our algorithm when provided synthetic observations generated on a preference graph $\mathcal{P}$ with fixed numbers of customers and items. Our results show that as the average customer degree decreases, the misclassification rate of our algorithm increases, as expected. However, the error increases at a much smaller rate compared to a standard latent class (LC) benchmark, which classifies customers using the estimated posterior membership probabilities in different segments (details below). Specifically, our results show that compared to the LC benchmark, our approach is (1) 28% more accurate in recovering the true customer segments and (2) faster, with average 17× speedup in computation time.

Setup. We chose $m = 2,000$ customers and $n = 100$ items and considered the following standard latent class generative model: The population consists of K customer segments with q_k denoting the proportion of customers in segment k; we have $q_k > 0$ for all $k \in [K]$ and $\sum_{k=1}^{K} q_k = 1$. The preference graph is generated according to the standard Erdős-Rényi (Gilbert) model with parameter $0 < p < 1$: each edge (i, j) between customer i and item j is added independently with probability p. The parameter $1 - p$ quantifies the *sparsity* of the graph: higher the value of $1 - p$, sparser the graph. All customers in segment $k \in [K]$ generate binary labels as follows: given parameter $\alpha_k \in (0, 1)$, they provide rating $+1$ to item j with probability α_k and rating -1 with probability $1 - \alpha_k$.

We denote each ground-truth model type by the tuple: $(K, 1-p)$. We generated 15 models by varying K over the set $\{5, 7, 9\}$ and $1 - p$ over the set $\{0, 0.2, 0.4, 0.6, 0.8\}$. For each value of K, we sampled the segment proportions from a Dirichlet distribution with parameters $\beta_1 = \beta_2 = \cdots = \beta_K = K + 1$ which tries to ensure that all segments have sufficiently large sizes by placing a large probability mass on equal proportions. Similarly, for each K, the

parameters α_k are chosen as $\alpha_k = 0.05 + 0.9(k-1)/(K-1)$ (i.e., K uniformly spaced points in the interval $[0.05, 0.95]$) for all $1 \le k \le K$.

For each ground-truth model type, we randomly generated 30 model instances as follows: (a) randomly partition the customer population into K segments with segment k having proportion q_k; (b) randomly generate the customer-item preference graph by adding edge (i, j) between customer i and item j with probability p; and (c) for each edge (i, j) in the preference graph, assign rating $+1$ with probability α_k and -1 with prob. $1 - \alpha_k$ where customer i belongs to segment k.

LC benchmark. Given the preference graph $\mathcal{P}$ and associated ratings $\boldsymbol{x}_1^{\text{obs}}, \boldsymbol{x}_2^{\text{obs}}, \cdots, \boldsymbol{x}_m^{\text{obs}}$, the LC method estimates the model parameters via MLE (see Appendix A.2 for the details). To ensure that the results for the LC benchmark are robust to how the parameters are estimated, we solved the MLE problem using two different methods: (a) the popular EM algorithm and (b) Sequential Least Squares Programming (SLSQP)[4]—a standard off-the-shelf solver. After the model parameters are obtained, customers are assigned to the segment for which the posterior probability of membership is the largest. Note that the LC method estimates a total of $2 \cdot K$ parameters.

Model-based embedding algorithm. We compute the embedding scores of the customers using Algorithm 3. The pooled model f_{pool} is described by a single parameter:

$$\alpha_{\text{pool}} = \frac{\sum_{i=1}^{m} \sum_{j \in N(i)} \mathbf{1}[x_{ij} = +1]}{\sum_{i=1}^{m} |N(i)|}.$$

We then cluster the embedding scores using the k-means algorithm to obtain the segments, and call our approach α-EMBED.

Results and Discussion. We measure the quality of the recovered clusters in terms of *accuracy*, defined as

$$\text{Accuracy}^{\mathsf{algo}} = 100 \times \left(\frac{1}{m} \sum_{i=1}^{m} \mathbf{1}[\hat{z}_i^{\mathsf{algo}} = z_i] \right),$$

where z_i is the true segment of customer i, $\hat{z}_i^{\mathsf{algo}}$ is the segment label assigned by method algo, and $\mathsf{algo} \in \{\text{LC}, \alpha\text{-EMBED}\}$. We label the true segments such that $\alpha_1 < \alpha_2 < \cdots < \alpha_K$. Then, for the LC method, we assign the segment labels in order of the estimated alpha parameters $\hat{\alpha}_k$, so that $\hat{\alpha}_1 < \hat{\alpha}_2 < \cdots < \hat{\alpha}_K$. For the α-EMBED method, recall from Lemma 1.1 that the embedding scores of customers in segment k concentrate around $H(\alpha_k, \alpha_{\text{pool}})/H(\alpha_{\text{pool}})$. Since $H(\alpha_k, \alpha_{\text{pool}}) = -\alpha_k \log \frac{\alpha_{\text{pool}}}{1-\alpha_{\text{pool}}} - \log(1 - \alpha_{\text{pool}})$, it follows that $H(\alpha_k, \alpha_{\text{pool}})$ is either increasing or decreasing in α_k depending on whether $\alpha_{\text{pool}} < \frac{1}{2}$ or $> \frac{1}{2}$. Therefore, we assign the segment labels in the increasing (resp. decreasing) order of the customer embedding scores when $\alpha_{\text{pool}} < \frac{1}{2}$ (resp. $\alpha_{\text{pool}} > \frac{1}{2}$).

Table 1.1 reports the accuracy of the LC and α-EMBED methods. Since there is no model misspecification, the LC method is statistically optimal and we see that it is able to recover the

[4]Specifically, we used Python SciPy library's `minimize` interface: https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html#scipy.optimize.minimize

Percentage accuracy in recovering true segments for different parameter settings

		LC			% IMPROVEMENT		AVERAGE SPEEDUP (X)	
K	$1-p$	EM	SLSQP	α-EMBED	over EM	over SLSQP	over EM	over SLSQP
5	0.0	99.0	99.0	98.7	-0.3^{**}	-0.3^{**}	20	251
	0.2	98.1	97.9	97.5	-0.6^{**}	-0.4^{**}		
	0.4	96.2	96.0	95.1	-1.1^{**}	-0.9^{**}		
	0.6	91.8	91.3	89.2	-2.8^{**}	-2.3^{**}		
	0.8	75.7	79.2	75.9	0.3	-4.2^{**}		
7	0.0	92.9	92.7	92.1	-0.9^{**}	-0.6^{**}	16	216
	0.2	89.3	89.4	88.3	-1.1^{**}	-1.2^{**}		
	0.4	82.5	84.1	83.1	0.7	-1.2^{**}		
	0.6	71.9	72.2	74.6	3.8^{**}	3.3^{*}		
	0.8	54.2	49.2	61.4	13.3^{**}	24.8^{**}		
9	0.0	72.5	81.2	80.8	11.4^{**}	-0.5	15	324
	0.2	65.7	71.2	75.6	15.1^{**}	6.2^{**}		
	0.4	58.3	58.1	70.4	20.7^{**}	21.2^{**}		
	0.6	47.9	47.9	61.5	28.4^{**}	28.4^{**}		
	0.8	39.1	35.2	49.1	25.6^{**}	39.5^{**}		

Table 1.1: The parameters are K—number of customer segments and $(1-p)$—sparsity of the preference graph. Each observation above is an average over 30 experimental runs. ** 1% and * 5% significance level according to a paired samples t-test.

true customer segments accurately when the preference graph is dense, but its performance suffers as the sparsity, $1-p$, increases. As sparsity increases, the number of data points per customer decreases, so the LC method encounters insufficient data to reliably estimate the $2K$ model parameters, particularly for larger values of K. The α-EMBED method, on the other hand, has comparable performance when there is enough data relative to the number of parameters being estimated. But it significantly outperforms the LC benchmark, by upto 28%, as the level of sparsity increases. The reason is that since the α-EMBED method estimates only a single parameter, it can make more efficient use of available data to determine the true segments. We also note that the performance improvement of α-EMBED over the LC benchmark is robust to the specific optimization method used for estimating the parameters of the LC model, be it EM or SLSQP.

Finally, as reported in Table 1.1, the fact that we estimate only a single model as opposed to K models results in an average 17× speedup compared to the EM method,[5] which is also sensitive to the initialization of the model parameters. This speedup becomes more significant when we have millions of customers and items, such as in our case-study at eBay (see Section 1.6), where the LC method is too computationally expensive and becomes infeasible to implement in practice.

1.5 Case study 1: Cold start recommendations in MovieLens dataset

Using the popular `MovieLens` [HKBR99] dataset, we now illustrate how our segmentation methodology can be applied to solve the classical cold-start problem in recommendation systems. This problem involves recommending *new* movies to users.[6] It is challenging because, by definition, new movies do not have any existing ratings. Yet, we need to account for heterogeneity in user preferences and personalize the recommendations. The baseline approach assumes that the population has homogeneous preferences and recommends the same set of movies to all the users. Compared to this benchmark, we show that segmenting the user population using our approach and customizing the recommendations to each segment can result in upto 48%, 55% and 84% improvements in the recommendation accuracy for drama, comedy, and action movies, respectively. In addition, compared to standard benchmarks used for capturing heterogeneity, our method outperforms: (1) latent class (LC) method by 8%, 13% and 10% and (2) empirical bayesian (EB) technique by 19%, 12% and 8%, respectively for the three genres. In addition, we achieve 20× speedup in the computation time over the LC benchmark.

Data Processing. The `MovieLens` dataset consists of 1M movie ratings (on 1-5 scale) from $6,040$ users for $3,952$ movies. For our analysis, we choose the three genres with the most number of movies in the dataset—drama, comedy, and action. We pose the new

[5] SLSQP takes significantly longer to converge, resulting in an average 264× speedup.

[6] To be consistent with the standard terminology in this problem context, we refer to customers as "users" in the remainder of the chapter.

movie recommendation task as the following prediction problem: given a movie, what is the probability that the user *likes* the movie? We say that a user *likes* a movie if the rating for the movie is greater than or equal to the average rating of the user across all the movies she rated, and *dislikes* the movie otherwise. Since the prediction task is only concerned with a binary (like/dislike) signal, we transform the raw user ratings to a binary like ($+1$) and dislike (-1) scale. Therefore, the preference graph consists of users on the left, movies on the right and the binary like/dislike ratings representing the edges.

We solve the prediction problem separately for each genre since user preferences can vary across different genres (see Appendix A.3.2 for the case when movies of different genres are considered together, where we also showcase the application of our technique when the model generating user ratings is only partially specified as discussed in Section 1.2.2). Further, since our goal is to recommend new movies to users (which have no or only a few ratings in practice), we select movies, for each genre, that have been rated by at least 30 users as part of the training set, and all others as part of the test set. Statistics for the training and test datasets are given in Table 1.2.

Methods and Metrics. The cold-start problem [SPUP02] has been studied extensively in the recommendation systems literature with solutions utilizing user-level and item-level attributes [PC09, ZTZX14] as well as social connections such as Facebook friends/likes or Twitter followers [LSKC13, SSB+14]. For our case study, the only information we use are the (transformed) user ratings and the genre of the movies. Consequently, existing collaborative filtering techniques are not directly applicable. In addition, the preference graph contains many missing ratings[7] and therefore, existing similarity-based clustering techniques that compute a similarity measure between users perform poorly (refer to Appendix A.3.1 for concrete numbers). However, our segmentation approach is precisely designed to handle sparsity in user observations and make it a natural choice in this context.

Recall that the goal is to predict the probability that a user gives $+1$ rating to a movie. To determine the benefits of segmentation for solving this prediction problem, we contrast two approaches—(1) *population model* : the user population is homogeneous so that all users have the same probability α for liking any movie; and (2) *segmentation model* : the population is composed of K segments, such that users in segment k have probability α_k of liking any movie. We first estimate the model parameters in both approaches using the training data and then, based on the estimated parameters, predict the ratings given by each user for movies in the test set. Let U denote the set of all users, and $N^{\text{train}}(i)$ and $N^{\text{test}}(i)$ denote respectively the set of movies in the training and test set rated by user i. For the *population model* approach, which we call POP, the maximum likelihood estimate (MLE) for parameter α is obtained by pooling all the ratings:

$$\alpha_{\text{pool}} = \frac{\sum_{i \in U} \sum_{j \in N^{\text{train}}(i)} \mathbf{1}[r_{ij} = +1]}{\sum_{i \in U} |N^{\text{train}}(i)|},$$

where r_{ij} is the rating given by user i for movie j. For the *segmentation model* approach, we use our model-based embedding technique α-EMBED, described earlier in Section 1.4, which

[7] The average sparsity, i.e. # edges/ (# users · # movies) is $\sim 7\%$; see Table 1.2.

computes user embeddings based on the estimated pooled model α_{pool} and then clusters the embeddings using k-means to obtain the segments. Then, we compute each segment parameter as

$$\alpha_k^{\text{EMBED}} = \frac{\sum_{i \in U} \mathbf{1}[\hat{z}_i^{\text{EMBED}} = k] \cdot \left(\sum_{j \in N^{\text{train}}(i)} \mathbf{1}[r_{ij} = +1] \right)}{\sum_{i \in U} \mathbf{1}[\hat{z}_i^{\text{EMBED}} = k] \cdot |N^{\text{train}}(i)|}$$

where $\hat{z}_i^{\text{EMBED}} \in [K]$ represents the assigned segment label for user i.

Given the above, the prediction for user i and new movie j_{new} is carried out as follows:

1. For the POP method, $\hat{r}_{ij_{\text{new}}}^{\text{POP}} = +1$ if $\alpha_{\text{pool}} \geq 0.5$, else $\hat{r}_{ij_{\text{new}}}^{\text{POP}} = -1$.
2. For the α-EMBED method, $\hat{r}_{ij_{\text{new}}}^{\text{EMBED}} = +1$ if $\alpha_{\hat{k}}^{\text{EMBED}} \geq 0.5$, else $\hat{r}_{ij_{\text{new}}}^{\text{EMBED}} = -1$ where $\hat{k} = \hat{z}_i^{\text{EMBED}}$.

We also compare our approach to two standard benchmarks that capture heterogeneity in user preferences: the LC method and the EB approach [RAM05] commonly used in the marketing literature. Refer to Appendix A.3.1 for details on these benchmarks.

There are many metrics to evaluate recommendation quality [SG11]. Since we are dealing with binary ratings, a natural metric is *accuracy*, i.e. the fraction of ratings that are predicted correctly. More precisely, let U^{test} denote the set of all users in the test set, note that $U^{\text{test}} \subseteq U$. Then for each user $i \in U^{\text{test}}$, we compute the individual accuracy as:

$$\text{Accuracy}_i^{\mathsf{method}} = \frac{1}{|N^{\text{test}}(i)|} \sum_{j_{\text{new}} \in N^{\text{test}}(i)} \mathbf{1}[r_{ij_{\text{new}}} = \hat{r}_{ij_{\text{new}}}^{\mathsf{method}}],$$

where $\mathsf{method} \in \{\text{POP}, \text{LC}, \text{EB}, \text{EMBED}\}$. The aggregate accuracy is then computed as

$$\text{Accuracy}^{\mathsf{method}} = 100 \times \left(\frac{1}{|U^{\text{test}}|} \cdot \sum_{i \in U^{\text{test}}} \text{Accuracy}_i^{\mathsf{method}} \right).$$

In the same manner, we can also compute the aggregate accuracy for a given segment k of users (identified by the α-EMBED method):

$$\text{Accuracy}_k^{\mathsf{method}} = 100 \times \left(\frac{1}{|\{i \in U^{\text{test}} : \hat{z}_i^{\text{EMBED}} = k\}|} \sum_{i \in U_{\text{test}} : \hat{z}_i^{\text{EMBED}} = k} \text{Accuracy}_i^{\mathsf{method}} \right)$$

Results and Discussion. Figure 1.1 shows the kernel density estimate of the user embeddings computed by our method for each genre. As noted in Section 1.2.3, our approach provides data-driven guidance on choosing a set of possible numbers of segments based on the number of modes in the estimated density; we try values of $K \in \{2, 3, 4, 5\}$ and choose the one that maximizes accuracy on a *validation set*[8]—a subset of the training data consisting

[8]This is standard practice in the machine learning literature; see for instance Section 4.3 in [AMMIL12].

Density of user embedding scores for different genres—Left: Action, Center: Comedy, Right: Drama

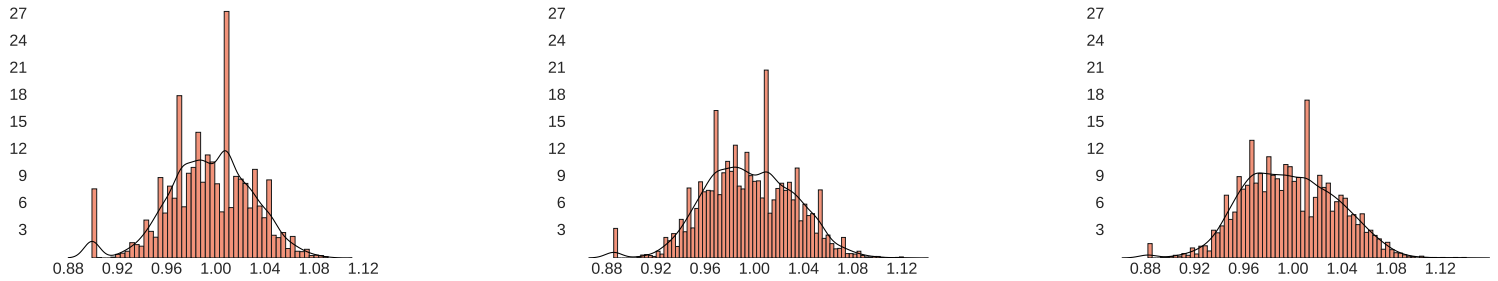

Figure 1.1: The x-axis corresponds to the embedding scores and the y-axis represents the number of users.

Training/test data statistics and aggregate rating prediction accuracy for the different genres

	Train Data			Test Data		Accuracy			
						Benchmarks			
Genre	Users	Movies	Ratings	Movies	Ratings	POP	EB	LC	α-EMBED
Action ($K = 2$)	6,012	453	257K	42	403	30.7	47.6	51.2	56.4
Comedy ($K = 4$)	6,031	945	354K	218	2,456	37.7	52.4	51.8	58.4
Drama ($K = 4$)	6,037	1,068	350K	425	4,627	38.6	53.2	53.0	57.2

Table 1.2: The number in parentheses represents the number of segments determined for each genre.

of movies that have relatively "small" number of ratings, i.e. at most 50 ratings. After segmenting the users, we predict their ratings for new movies as outlined above and compute the accuracy metrics for the different approaches.

Table 1.2 reports the aggregate accuracy of all the methods for each of the genres. The benefits of segmentation can be seen across all the genres, with improvements upto 84% (for the action genre) in the prediction accuracies. The population model treats the preferences of all users as being the same and performs poorly since it ends up recommending the same set of movies to all the users. The segmentation model, on the other hand, makes different recommendations to the different user segments, and consequently performs significantly better. Furthermore, using our segmentation algorithm performs better than both the LC (upto 13% for the comedy genre) and EB (upto 19% for the action genre) methods. This suggests that a discrete form of heterogeneity, resulting from a "hard" separation of the users into distinct segments, is better than a continuous or "soft" form of heterogeneity (as considered by the benchmarks) for the cold-start recommendation problem. In addition, our method is upto 20× faster than the LC method[9] when the population is grouped into $K = 4$ segments, again highlighting the fact that our algorithm is fast and can scale better to larger datasets.

To understand where the accuracy improvements come from, Table 1.3 displays the accuracy of the POP and α-EMBED methods, broken down by individual user segments computed by the α-EMBED method. Also shown are the estimated pooled model α_{pool} and segment parameters α_k^{EMBED}. Now observe that for segments 1 & 4 in the drama and comedy genres, the estimated parameters α_k^{EMBED} are furthest from the pooled parameter α_{pool}. In other words, these segments contain users whose preferences are very different from that of the population, i.e. *esoteric* preferences, which are not captured well by the pooled model α_{pool}. Using the segment parameters α_k^{EMBED} for the rating predictions results in significant improvements in the accuracy for segment 4 users—upto 1.8× and 2.5× increase for the

[9]We use the EM algorithm to estimate the model parameters.

Comparison of rating prediction accuracy of population model and our model-based embedding technique by individual user segments

Genre	α_{pool}	User Segments	α_k^{EMBED}	$\text{Accuracy}_k^{\text{POP}}$	$\text{Accuracy}_k^{\text{EMBED}}$	% increase
Action	0.537	Segment 1 (2,900)	0.658	35.6	35.6	—
		Segment 2 (3,112)	0.441	26.9	73.1	171.7
Comedy	0.543	Segment 1 (941)	0.749	45.2	45.2	—
		Segment 2 (2,062)	0.631	45.9	45.9	—
		Segment 3 (1,869)	0.495	33.2	66.8	101.3
		Segment 4 (1,159)	0.360	26.4	73.6	178.7
Drama	0.545	Segment 1 (1,164)	0.736	50.1	50.1	—
		Segment 2 (1,975)	0.620	43.5	43.5	—
		Segment 3 (1,769)	0.485	36.1	63.9	77.0
		Segment 4 (1,129)	0.342	22.5	77.5	244.4

Table 1.3: The numbers in parentheses represent the size of each user segment. "% increase" denotes the percentage improvement in accuracy of our segmentation approach over the population model.

comedy and drama genres respectively. Note that we do not observe any improvement for segment 1 users, this is because of our experimental setup which involves a coarse-grained rating prediction based on a threshold of 0.5 (see "Methods and Metrics" above). The users in the intermediate segments 2 & 3, on the other hand, have preferences that are very similar to those of the population, i.e *mainstream* preferences. However, we are still able to distinguish between users in these segments resulting in improved rating prediction accuracy for segment 3 users. The improvements are lower than for the esoteric segments, since the pooled model is already able to capture the preferences of the mainstream users. The story is similar for the action genre where segment 1 (resp. 2) behaves as the mainstream (resp. esoteric) segment.

1.6 Case study 2: Personalized Recommendations on eBay

In this section, we use our segmentation methodology to *personalize* similar product recommendations on eBay. When a user is on a product page, eBay recommends products that are "similar" to the product being viewed (the *seed* product); see Figure 1.2 for an example. The recommended products are shown below the seed product, but above the fold.[10] Even without personalization, determining similar products is a challenging task because the

[10] Above the fold refers to the portion of the webpage that is visible without scrolling.

Example of similar product recommendation on eBay

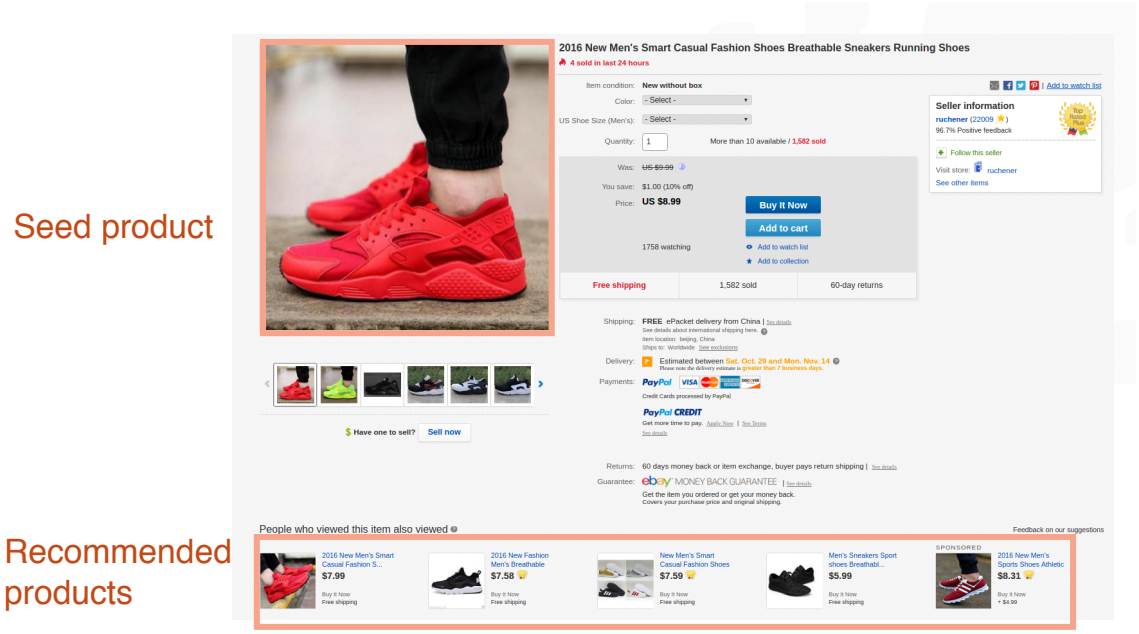

Figure 1.2: The seed product is a pair of sneakers that the user is currently viewing, and the recommended products at the bottom are other pairs of shoes similar to the seed product.

product listings offered on the eBay marketplace are diverse. They range from Fitbit tracker or iPhone (products with well-defined attributes) to obscure antiques and collectibles that are highly unstructured, and they can also have multiple conditions—new, used, refurbished, etc.—and selling formats—fixed price vs auction. In addition, many products tend to be short-lived—they surface on the site for one week and are never listed again. Nevertheless, [BJS+16] were able to address these challenges in a scalable recommendation system they implemented at eBay. While their approach resulted in positive lift in critical operational metrics, it does not take into account heterogeneous user preferences—for a given seed product *every* user is recommended the *same* set of products. Personalizing the recommendations to individual users is however challenging because most users interact with only a small fraction of the catalog—in our sample dataset below, a user on average interacted with only 5 out of ~ 4.2M items. Such sparsity makes it hard to determine individual preferences and limits the application of traditional collaborative filtering algorithms. We are able to use our segmentation methodology to address this challenge. We show that segmenting the user population using our technique and personalizing the recommendations to each segment can result in upto 8% improvement in the recommendation accuracy. It matters how we obtain the segments because other natural approaches to segmentation that are based on similarity in demographics (age, gender, income, etc.) and aggregate purchase behavior resulted at best in only $\sim 1\%$ improvement.

Data. The raw data consist of impressions collected over a two-week period in the summer

of 2016. An impression is generated whenever a user interacts, either by clicking or clicking and then purchasing, with a recommended product on a product page. The impression contains information on the user, the seed product, the recommended (reco) products, and the actions (clicks and purchases) taken by the user on the reco products. The data provided to us were a random subset of all impressions that had at least one click on a reco product. We transform these data into a preference graph as follows. For each impression, we extract the user and the set of (seed, reco) product pairs. We assign the users to the nodes on the left and the (seed, reco) pairs to the item nodes on the right. We add an edge from a user node to a (seed, reco) node if there is an impression in which the user has taken an action on the (seed, reco) pair, and assign a binary label to the edge: the $+1$ label if the user clicked and then purchased the reco product and the -1 label if the user did not purchase the reco product, irrespective of whether it was clicked or not. The resulting preference graph consists of $\sim$ 1M users, $\sim$ 4.2M items, i.e. (seed, reco) product pairs, and $\sim$ 4.5M edges across different product categories on eBay. For our analysis, we focus on the two categories with the most numbers of purchases—`Clothing, Shoes and Accessories (CSA)` and `Home & Garden (HG)`. In each category, we randomly split all the binary purchase/no-purchase signals into training (80%) and test (20%) sets. Table 1.4 reports the summary statistics of the data.

Current approach. [BJS+16] transformed the problem of generating similar recommendations into the following prediction problem: for a fixed seed product, what is the probability that a candidate reco product is purchased? The candidate recommendations are then ranked (in real-time) according to the probability of being purchased. The prediction problem was solved by learning a classifier on the above purchase/no-purchase user interaction data. In particular, they assumed that the population is homogeneous and therefore estimate a single logistic regression classifier[11] on the user interaction data.

The authors performed feature engineering to ensure the classifier had good performance; refer to [BJS+16] for more details. Let U denote the set of all users and $N^{\text{train}}(i)$ denote the set of items that user $i \in U$ has interacted with in the training data. Let $\boldsymbol{y}_{s,r} \in \mathbb{R}^D$ represent the feature vector corresponding to item (s, r) where s is the seed product and r is the reco product. Further, let $x_{i,(s,r)} \in \{-1, +1\}$ denote the binary purchase $(+1)$ or no-purchase (-1) signal associated with a user i that interacted with item (s, r). Then, logistic regression estimates a parameter vector $\boldsymbol{\omega}_{\text{pool}} \in \mathbb{R}^D$ by solving:

$$\min_{\boldsymbol{\omega}} \sum_{i \in U} \sum_{(s,r) \in N^{\text{train}}(i)} -\mathbf{1}[x_{i,(s,r)} = +1] \cdot \log p_{s,r}(\boldsymbol{\omega}) - \mathbf{1}[x_{i,(s,r)} = -1] \cdot \log\left(1 - p_{s,r}(\boldsymbol{\omega})\right),$$

where $p_{s,r}(\boldsymbol{\omega}) = \frac{\exp(\boldsymbol{\omega}^\top \boldsymbol{y}_{s,r})}{1+\exp(\boldsymbol{\omega}^\top \boldsymbol{y}_{s,r})}$ represents the probability of purchase signal on item (s, r) under parameter $\boldsymbol{\omega}$. We call this approach the *population model*, f_{pool}.

Our approach: capturing heterogeneity through segmentation. The extreme

[11] The authors tried out several different binary classifiers and observed that a logistic regression model achieves comparable performance to more sophisticated methods like random forests and boosted decision trees, and consequently, decided to pick the logistic regression classifier for its simplicity and scalability.

Density of user embedding scores for `CSA` (Left) and `HG` (Right) categories

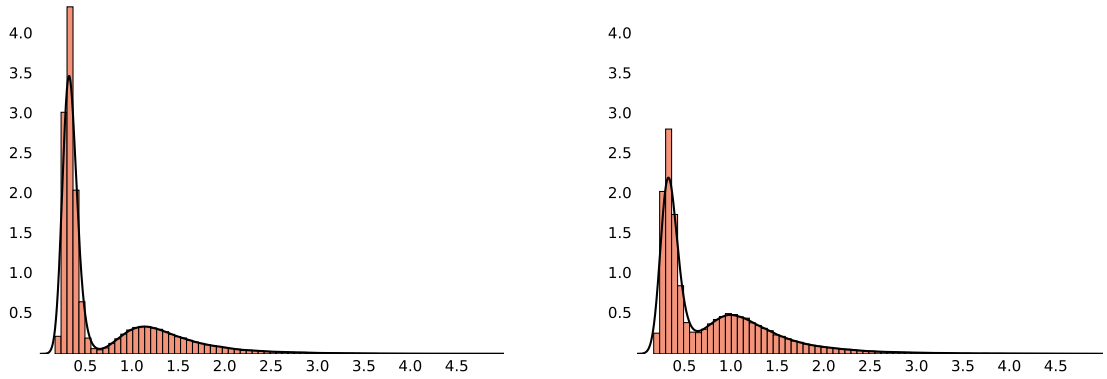

Figure 1.3: The x-axis, representing the user embedding scores, is cutoff at 5 so that the two modes above can be visualized better. The largest embedding score is 28.3 for the `CSA` category and 25.6 for the `HG` category

sparsity[12] of the preference graph limits the applicability of most existing clustering techniques. We applied our model-based embedding technique to segment the user population using the preference graph constructed above. The pooled model corresponds to the aforementioned logistic regression classifier f_{pool}; then we apply Algorithm 3 to compute the embeddings of each user i:

$$v_i = \frac{\sum_{(s,r)\in N^{\text{train}}(i)} -\mathbf{1}[x_{i,(s,r)} = +1] \cdot \log p_{s,r}(\boldsymbol{\omega}_{\text{pool}}) - \mathbf{1}[x_{i,(s,r)} = -1] \cdot \log\left(1 - p_{s,r}(\boldsymbol{\omega}_{\text{pool}})\right)}{\sum_{(s,r)\in N^{\text{train}}(i)} H\left(p_{s,r}(\boldsymbol{\omega}_{\text{pool}})\right)}$$

where $H(\cdot)$ is the binary entropy function introduced in Section 1.3.1. We perform k-means clustering on the embeddings to obtain the different segments. Once we obtain the segments, we estimate a separate parameter $\boldsymbol{\omega}_k$ for each segment k by fitting a logistic regression classifier f_k using only the interactions of users in segment k.

Results and Discussion. Figure 1.3 shows the kernel density estimate of the user embeddings computed by our algorithm. While one can visually see two modes in the density plots, we also observed a long tail and consequently, chose $K = 3$ segments to account for those users. Brovman et al. used the area-under-the-curve (AUC) metric[13] to evaluate the performance of the logistic regression classifier. For each user segment k, we compare the AUC of the population model f_{pool} with that of the segment-specific classifier f_k, on the test set. Table 1.4 shows the percentage improvement in AUC of the segment-specific classifiers across each category.

[12]The average sparsity in our dataset is $\sim 0.0007\%$ which is orders of magnitude smaller than in our MovieLens case study as well as the typical sparsity in Netflix-like rating systems [BK07].

[13]AUC measures classifier performance as equal to the likelihood that the classifier will rank (based on the probability of positive class label) a randomly chosen positive example higher than a randomly chosen negative example.

AUC improvements of segment-specific logistic regression classifiers over population model

Product	Train Data			Test Data				Time to
Category	Users	Items	Edges	Items	Edges	Segments	% AUC imp.	segment
CSA	172K	534K	561K	116K	118K	Segment 1 (120K)	-0.27	∼6 mins
						Segment 2 (46K)	0.54	
						Segment 3 (6K)	5.78	
HG	129K	432K	458K	100K	102K	Segment 1 (78K)	0.08	∼5 mins
						Segment 2 (45K)	-0.37	
						Segment 3 (6K)	7.95	

Table 1.4: The numbers in parentheses denote the size (in thousands) of each user segment.

We observe that our segmentation technique can lead to significant improvements in the AUC—upto 6% in category CSA and 8% in category HG. The improvements are the largest for segment 3 or the *esoteric* segment, which consists of users having the highest embedding scores. This, in turn, means that they deviate most from the population preferences and therefore the population model is not able to capture the preferences of these users. Segment 1 or the *mainstream* segment, consists of users having the lowest embedding scores; their preferences are captured well by the population model and therefore the benefit from segmentation is negligible.[14] Segment 2 consists of users who have "intermediate" preferences; they agree with the population on some items but deviate on other items. Their preferences are partially captured by the population model and depending on whether they have more mainstream or esoteric preferences, treating such users separately can result in loss (as for HG category) or gain (as for CSA category) in performance.

The above improvement obtained using our segmentation technique is non-trivial considering the fact that we also tried several natural approaches such as segmentation based on similarities in demographics (age, gender, income, etc.) and aggregate purchase statistics (number of transactions and/or amount spent in the last year, etc.). But the best of these resulted in around 1% improvement in AUC for any user segment, compared to the population model. Such approaches implicitly assume that similarity in demographics or aggregate purchase behavior implies similarity in preferences, which might not be the case in practice. Instead, focusing on fine-grained user interactions such as click and purchase signals can help to directly capture their preferences. However, a major challenge in using such data is that it is extremely sparse; for instance, in the dataset above, users had only 4-5 observations on average and consequently, most of the users do not have any overlap in the observations that they generate. This makes it hard to determine whether two users have similar preferences.

[14]For CSA, the AUC is (slightly) lower and we attribute this to having smaller number of samples in the training data for the segment-specific classifier.

Further, existing techniques like the LC method are prohibitively slow for such a large dataset. Table 1.4 also shows the time taken to segment the population using our technique, without any optimizations or parallel processing. We believe our implementation can be easily ported to large-scale distributed data processing frameworks like Apache Spark to obtain further speedups. This shows that our segmentation technique can scale to large datasets and work directly with fine-grained user observations (such as click and purchase signals) to generate personalized product recommendations.

Chapter 2

Segmenting crowd workers based on their reliability

2.1 Introduction

The growing popularity of online crowdsourcing services like Amazon Mechanical Turk and CrowdFlower has made it easy to collect low-cost labels from the crowd to generate training datasets for machine learning applications. Unfortunately, these labels are typically of low quality because of unintentional or intentional inaccuracies introduced by unreliable and malicious workers [KCS08, LEHB10]. Determining correct labels of tasks from such noisy labels is challenging because the reliabilities or qualities of workers are often unknown. While one may use "gold standard" tasks—whose true label is already known—to identify low-reliability workers [SOJN08, DHSC10, LEHB10], accessing true labels for a sufficient number of tasks can be difficult and expensive. To address these challenges, a common solution is to use *redundancy* [SPI08], that is, collecting multiple labels for each task and assigning multiple tasks to each worker. Given the redundant labels, most existing studies make specific probabilistic assumptions on how individual workers provide labels and propose techniques to infer true task labels given workers' responses. Common probabilistic models include the one-coin model [ZCZJ14], two-coin model [RY12], and the general Dawid-Skene model [DS79]. For example, the "one-coin" model assumes that each worker w provides the correct label to an assigned task with probability p_w and (an) incorrect label with probability $1 - p_w$. The parameter p_w thus measures the reliability of worker w.

While most existing work relies on explicit probabilistic models, recent studies [VdVE11, DDCM12] and anecdotal evidence show that worker labeling strategies may not be probabilistic in practice. For instance, for binary classification tasks, workers may adopt strategies that: (a) uniformly label all tasks $+1$ if it is known that $+1$ labels are more prevalent than -1 among true labels in the corpus of tasks;[1] (b) provide accurate labels to the first few tasks and random labels to remaining ones; and (c) systematically provide $+1$ labels to certain types of

[1]Note that the one-coin model cannot capture this strategy, but the more general two-coin model [RY12] can.

tasks and -1 to other types. [VdVE11] recently provided real-world empirical evidence of similar worker strategies. In addition, workers may be malicious and may adopt sophisticated strategies for explicitly altering inferred labels of tasks. For instance, [WWZZ14] showed that malicious crowdsourcing campaigns, called "crowdturfing", are increasing in popularity in both dedicated and generic crowdsourcing websites [MML+11, WWZ+12]. Further, evidence indicates the presence of malicious users in online content rating systems (such as Digg, Amazon, and Yelp) in which users can choose items to rate and the ratings are public. Specifically, users have been observed to explicitly alter the popularity of advertisements and phishing articles [TMLS09] and collaboratively target products on Amazon [MLG12].

This chapter focuses on the problem of explicitly identifying *unreliable* and *adversarial* workers in crowdsourced labeling tasks by using only the labels provided by workers as an input. We consider a general crowdsourcing setting in which users/workers provide labels to items/tasks. The setting may be a crowdsourced classification application (such as Mechanical Turk) in which labels are collected for tasks by assigning[2] them to workers; or a public crowdsourcing system (such as Digg, Amazon, or Yelp) in which users provide labels/ratings to a collection of items they choose. For brevity, we use the generic terms "worker" and "task" for both types of applications. We make the following assumptions. The tasks have binary true labels in the set $\{-1, +1\}$; for cases in which the notion of true task labels is subjective, we consider it to be the population's majority opinion. We distinguish between two types of workers: *honest* and *adversarial*. The population of workers is mostly honest, with adversaries comprising a "small" fraction. Honest workers provide responses according to a well-defined probabilistic model, say, $\hat{M}$ (e.g. the one-coin model introduced above), and therefore, they *can make mistakes*. However, the model $\hat{M}$ is not known. Motivated by the presence of non-random worker strategies, we go beyond standard probabilistic models and consider a much broader class of *adversarial* worker strategies. Specifically, adversaries adopt strategies different from those of honest workers, whether probabilistic or not, that is, their responses are not generated according to model $\hat{M}$. Further, different adversaries may adopt distinct strategies.

Our work is different from most prior literature [DS79, SFB+95, WRW+09, RYZ+10, WBBP10, GKM11, KOS11, LPI12, ZBMP12, DDKR13, ZCZJ14] that primarily focused on designing label aggregation algorithms to maximize the accuracy of recovered task labels. Only a few studies have explicitly focused on identifying unreliable workers based on their responses; these either relied on access to true task labels [SOJN08, DHSC10, LEHB10] or did not assume any form of adversarial behavior [VdVE11, RY12, HBKVH13]. We aim to identify unreliable and adversarial workers using *only* their responses and do not assume access to any true task labels. Furthermore, to the best of our knowledge, we are unaware of previous studies that have addressed the problem of identifying adversarial workers who can adopt *arbitrary* strategies in crowdsourcing systems. Refer to Section 2.1.1 for further discussion on this aspect.

For the above setting, we design a scoring algorithm that computes "reputation scores"

[2]Workers can still choose from among assigned tasks; however, the assignment can be done to ensure that the graph representing workers' assignment to tasks has particular structures—see Section 2.4.1.

for workers to indicate the degree to which their labeling patterns are adversarial. We base our algorithms on the intuition that as the population is mostly honest and adversaries' labeling patterns differ from those of honest workers, adversary labeling patterns should be statistical outliers. The reputation score then indicates the degree to which a worker's labeling pattern is a statistical outlier. The adversaries identified by our algorithms may be discarded or processed separately depending on the application. Section 2.5 shows that discarding adversary labels can enable standard label aggregation algorithms to infer true task labels more accurately.

We note that the problem of identifying unreliable honest workers and adversaries from only their responses is nontrivial, especially because we cannot access true labels for any of the tasks. In particular, even in the simple and commonly observed case where adversaries always provide the label $+1$, the natural approach of classifying workers who only give $+1$ labels as adversaries can have arbitrarily bad performance (see Lemma 2.1). Furthermore, because we do not restrict the adversary strategy in any way, differentiating them from honest workers can be difficult. In fact, we show (see Theorem 2.10) that by carefully choosing their responses, *sophisticated* adversarial workers can render themselves *indistinguishable* from honest ones and ensure that the true labels of some fraction of tasks cannot be inferred better than a random guess.

2.1.1 Related Work

Our work is part of the literature on crowdsourcing that proposes statistical techniques to exploit the redundancy in collected labels to simultaneously infer the latent reliabilities of workers and true task labels. In particular, our work is related to three broad streams. The first stream focuses on "crowdsourced classification", namely, inferring underlying true labels of tasks when workers adopt specific probabilistic labeling strategies. Our reputation algorithm can work in conjunction with any of these methods, possibly by filtering out low reputation workers. The second stream proposes methods to explicitly filter out low-reliability workers; it is similar in spirit to our approach. The third stream focuses on methods to address sophisticated attacks in online settings; it is related to our treatment of sophisticated adversaries.

Crowdsourced classification. The literature on crowdsourced classification is vast. Most studies are based on the worker model proposed by [DS79], which is a generalization of the one-coin model to tasks with more than two categories. The standard solution is to use the expectation-maximization (EM) algorithm (or its variants) to estimate worker reliability parameters and true task labels [SFB+95, RYZ+10]. The methods proposed in [LPI12] and [CLZ13] take a Bayesian approach by assuming different priors over the worker reliability parameters. [WRW+09] included task difficulty as an additional parameter in the model, and [WBBP10] studied a model with multi-dimensional latent variables for each worker, such as competence, expertise, and bias. [ZBMP12, ZLP+15] introduced a natural generalization of the Dawid-Skene model that captures tasks with differing difficulties and proposed a minimax entropy based approach which works well in real datasets. Although most of these approaches

show improved performance on real-world datasets, they offer no theoretical guarantees on the resulting estimates of true task labels and model parameters.

From a theoretical perspective, the crowdsourced classification problem has been studied in two distinct regimes: *dense* and *sparse*. In the *dense* regime, it is assumed that each worker has a certain probability of labeling each task. As the problem size (or number of workers) increases, each task receives an increasing number of responses; therefore, the true labels of all tasks are eventually identified correctly (with high probability). The theoretical analysis therefore focuses on identifying the rate at which the algorithm estimates converge to the true task labels under different settings. The dense regime was first studied by [GKM11], who proposed a spectral method to infer task labels. More recently, [GZ16] studied the minimax optimal error rate of a projected EM algorithm under the one-coin model, and [LY14] provided upper bounds on the error rate of weighted majority voting algorithms for the Dawid-Skene model. [ZCZJ14] showed that the EM algorithm for the Dawid-Skene model achieves the optimal convergence rate when initialized using a spectral method.

In the *sparse* regime, each task is assigned a "small" number of workers, that is, of size $O(1)$, so that the accuracy does not increase with problem size. [KOS14] were the first to analyze this scenario; they proposed an iterative message-passing algorithm for estimating true task labels as well as a task assignment scheme that minimizes the total price that must be paid to achieve an overall target accuracy. They showed that their algorithm is optimal by comparing it against an oracle estimator that knows the reliability of every worker. [DDKR13] proposed methods based on singular-value decomposition (SVD) and analyzed the consistency of their estimators for the one-coin model. Recently, [KO16] analyzed the "Generalized Dawid-Skene Model" introduced by Zhou et al. and showed that spectral approaches achieve near-optimal performance, whereas [OOSY16] proved that belief propagation (BP) is optimal, that is, it matches the performance of the MAP estimator, under the one-coin model when each worker is assigned at most two tasks. In our theoretical analysis, we focus on this regime. The key distinction of our work from previous studies is that we focus on characterizing the misclassification rate of our algorithm in classifying workers, as opposed to the accuracy of recovering true task labels.

Detecting Unreliable/Adversarial workers. Some studies have aimed to explicitly detect and/or remove unreliable workers based on observed labels. One approach is to use "gold standard" tasks, that is, tasks whose true labels are already known, to identify low-reliability workers [SOJN08, DHSC10, LEHB10]. However, accessing true task labels can be difficult and might involve additional payment. In this work, we do not assume access to any gold standard tasks and identify adversarial workers based only on the provided labels. [VdVE11] defined scores customized to specific adversary strategies to identify and remove them. Similarly, [HBKVH13] modeled the worker population as consisting of two types—those who always provide the correct label and spammers who provide uniformly random labels—and estimated each worker's trustworthiness by using the observed labels. In contrast to these studies, we allow adversaries to adopt arbitrary strategies. [IPW10] proposed a method for quantifying worker quality by transforming the observed labels into

soft posterior labels based on the estimated *confusion matrix* [DS79]. Similar to our work, their approach computes an expected cost for each worker, where the higher the cost, the lower is the worker's quality. [RY12] proposed an empirical Bayesian algorithm to eliminate workers whose labels are not correlated with the true label (called *spammers*), and estimated consensus labels from the remaining workers. Both of these works rely on the Dawid-Skene model, whereas our algorithms do not assume knowledge of the probabilistic model used by workers. Some studies have tried to quantify the price of having adversarial workers under some restricted settings. [GKM11] (in the dense regime) and [KOS13] (in the sparse regime) considered malicious workers who can collude and provide arbitrary responses to degrade the performance of the aggregation algorithms and showed that their approaches are robust to manipulation by a small constant fraction of such adversaries. However, both these works assume specific structures on the worker-task assignment graph and do not consider adversaries who can adapt their responses based on the labels submitted by honest workers. Our analysis considers *arbitrary* worker-task assignment graphs, and we allow adversaries to choose their labels based on observed honest workers' responses.

Sybil attacks. Finally, our work is also broadly related to the rich literature on identifying *Sybil* identities in online social networks. Most such schemes [YKGF06, YGKX08, DM09, TLSC11, VMG+12] use the graph (or trust) structure between users to limit the corruptive influences of *Sybil attacks* (see [VPGM10] for a nice overview). In our context, there is no information about the network structure or trust relationships between workers, and because most crowdsourcing tasks involve some form of payment, it is harder to launch Sybil attacks by forging financial credentials like credit cards or bank accounts.

2.2 Problem Setup

We consider the following broad setting. There is a set $\mathcal{T} = \{t_1, t_2, \ldots, t_m\}$ of m tasks such that each task t_j is associated with a latent ground-truth binary label $y_j \in \{-1, +1\}$. We elicit binary labels for these tasks from a set $W = \{w_1, w_2, \ldots, w_n\}$ of n workers. Each worker typically labels only a subset of the tasks, and we generically say that the subset of tasks is *assigned* to the worker. We represent this assignment by using a bipartite graph $\mathcal{B} = (W \cup \mathcal{T}, E)$ with workers on one side, tasks on the other side, and an edge $(w_i, t_j) \in E$ indicating that task t_j is assigned to worker w_i. We call $\mathcal{B}$ the *worker-task assignment graph* and suppose that the assignment is pre-specified.

Each worker w_i provides a binary response[3] $w_i(t_j) \in \{-1, +1\}$ for each task t_j assigned to it. We encode the responses as the *response matrix* $\mathcal{L} \in \{-1, 0, +1\}^{|W| \times |\mathcal{T}|}$ such that $\mathcal{L}_{ij} = w_i(t_j)$, for all $1 \leq i \leq n$ and $1 \leq j \leq m$, where we set $w_i(t_j) = 0$ for any task t_j not assigned to worker w_i. $W_j \subseteq W$ denotes the set of workers who labeled task t_j and $\mathcal{T}_i \subseteq \mathcal{T}$, the set of tasks assigned to worker w_i. Let d_j^+ (resp. d_j^-) denote the number of workers labeling task t_j as $+1$ (resp. -1).

[3] We use the terms "label" and "response" interchangeably.

Worker model. We assume that the population of workers comprises two disjoint classes: *honest* and *adversarial*. That is, $W = H \cup A$ with $H \cap A = \emptyset$, where H is the set of honest workers and A, the set of adversarial workers. The class memberships of the workers are latent, so we do not know whether a worker is honest. Honest workers provide (noisy) labels according to an unknown but explicit probabilistic model (such as the one-coin model). Adversarial workers are those whose labeling strategy does not conform to this probabilistic model; they can adopt arbitrary (deterministic or probabilistic) strategies. If honest workers adopt the one-coin model, example adversary strategies include (a) the *uniform strategy*, in which the worker arbitrarily provides uniform responses $+1$ or -1 to all assigned tasks irrespective of the true label; (b) *smart strategy*, in which the adversary is smart and chooses the uniform label in accordance with the population prevalence of the true labels so that if more than 50% of tasks are a priori known to have label -1, then the worker chooses -1 as the uniform label and vice-versa; or (c) *sophisticated strategy*, in which the worker adopts strategies specifically designed to cause the maximum "damage" (see Section 2.4.2 for details).

Most existing works only focus on honest workers, whereas our approach also considers adversarial workers. Furthermore, our definition of "adversary" is intentionally broader than common definitions to accommodate a wide range of labeling strategies. In fact, some of the abovementioned example adversary strategies may be accommodated by extensions of the one-coin model; for example, the uniform strategy is captured by the two-coin model [RY12], and the smart strategy can be accommodated by allowing worker reliability parameters to depend on the population prevalence of the task labels. While such case-by-case extensions are feasible in theory, they do not extend to general adversary strategies, including sophisticated strategies specifically designed to inflict the maximum "damage".

Given the broad definition of adversaries, our approach is to design a general algorithm to *identify* adversarial workers, given access to only the response matrix $\mathcal{L}$. The adversaries identified using our algorithm may be filtered out or investigated further, depending on the application. We describe a reputation-based algorithm that only relies on the response matrix $\mathcal{L}$ to adversaries. The algorithm relies on detecting workers whose labeling patterns are statistical *outliers* among the population of workers.

2.3 Reputation Algorithms

We now describe the proposed algorithm for identifying adversarial workers given the response matrix $\mathcal{L}$. We suppose that no side information is available on the workers' identities (e.g., a social network or worker-level demographic information), and therefore, the algorithm must solely rely on the response patterns given by workers. Our approach is to compute a "reputation" or "trust" score for each worker as a measure of the degree to which their response pattern is a statistical outlier or an anomaly. Workers with low reputation scores are significant outliers and are identified as adversaries.

To compute a worker's reputation, the algorithm relies on the number of *conflicts* the

worker is involved in. A worker is involved in a conflict if its response to an assigned task is in disagreement with those of other workers. Note that tasks with a consensus opinion, that is, those having all $+1$ or -1 labels, do not provide any *discriminative* information about the workers who labeled the task. In other words, we cannot distinguish between honest and adversarial workers from just this specific task. Therefore, we focus on tasks that lack consensus, that is, those having a mix of $+1$ and -1 labels. We call this subset of tasks the *conflict set* $\mathcal{T}_{cs}$, and workers who respond to tasks in this set are all involved in conflicts. A conflict typically indicates the presence of low-reliability honest workers (who tend to make mistakes) or adversaries. In the ideal case, when all honest workers are perfectly reliable, a conflict necessarily means the presence of an adversarial worker. In this case, the number of conflicts a worker is involved in can serve as a rough indicator of the possibility of this worker being an adversary. However, when honest workers are not perfect and make mistakes, a conflict indicates only a *chance* of the presence of an adversary. Then, simply counting the number of conflicts may over-penalize honest workers who label a large number of tasks.

To overcome this issue, we propose two penalty allocation techniques, resulting in two variants of our algorithm: (a) *soft-penalty* and (b) *hard-penalty*.

2.3.1 Soft-Penalty Algorithm

In the *soft-penalty* algorithm (see Algorithm 5), for any task t_j in the conflict set, we allocate a penalty of $1/d_j^+$ and $1/d_j^-$ to all workers who provide the label $+1$ and -1 for t_j, respectively. Then, for each worker, we compute the *net penalty* by averaging the penalties across all assigned (conflict) tasks.

The above allocation of penalties implicitly rewards agreements among worker responses by making the penalty inversely proportional to the number of other workers that agree with a worker. In particular, if a worker agrees with the majority opinion on some task, then it is allocated a lower penalty than a worker who disagrees with the majority. Further, averaging normalizes for the number of tasks labeled by any worker. The algorithm relies on the following intuition for allocating penalties: assuming the average reliability of honest workers to be $> \frac{1}{2}$, we expect that honest workers provide the correct response to the assigned tasks on average. Furthermore, because there are more honest workers than adversaries, we expect the majority response to be the same as the true label of the task for most tasks. Therefore, we expect that the above allocation of penalties assigns lower penalties to high-reliability honest workers and higher penalties to low-reliability honest *and* adversarial workers. We formalize this intuition in Section 2.4.1, where we prove theoretical guarantees for the soft-penalty algorithm. We show that the soft-penalty algorithm performs well in identifying low-reliability honest workers as well as adversarial workers employing deterministic strategies (see Theorems 2.4 and 2.7). Our results demonstrate the asymptotic consistency of the soft-penalty algorithm in identifying adversaries under standard assumptions on the structure of the worker-task assignment graph.

Algorithm 5 SOFT-PENALTY

1: **Input:** W, $\mathcal{T}$, and $\mathcal{L}$

2: For every task $t_j \in \mathcal{T}_{cs}$, allocate penalty s_{ij} to each worker $w_i \in W_j$ as follows:

$$s_{ij} = \begin{cases} \frac{1}{d_j^+}, & \text{if } \mathcal{L}_{ij} = +1 \\ \frac{1}{d_j^-}, & \text{if } \mathcal{L}_{ij} = -1 \end{cases}$$

3: **Output:** Net penalty of worker w_i:

$$pen(w_i) = \frac{\sum_{t_j \in \mathcal{T}_i \cap \mathcal{T}_{cs}} s_{ij}}{|\mathcal{T}_i \cap \mathcal{T}_{cs}|}$$

Algorithm 6 HARD-PENALTY

1: **Input:** W, $\mathcal{T}$, and $\mathcal{L}$

2: Create bipartite graph $\mathcal{B}^{cs}$ as follows: (i) Each worker $w_i \in W$ is represented by a node on the left (ii) Each task $t_j \in \mathcal{T}_{cs}$ is represented by two nodes on the right, t_j^+ and t_j^- (iii) Add the edge (w_i, t_j^+) if $\mathcal{L}_{ij} = +1$ or edge (w_i, t_j^-) if $\mathcal{L}_{ij} = -1$

3: Compute optimal semi-matching $\mathcal{M}$ on $\mathcal{B}^{cs}$

4: **Output:** Net penalty of worker w_i:

$$pen(w_i) = deg_{\mathcal{M}}(w_i)$$

Although the soft-penalty algorithm can successfully identify adversarial workers adopting certain types of strategies, its performance depends on the complexity of these strategies. If adversarial workers are non-colluding and adopt non-deliberate strategies, then the soft-penalty algorithm can identify them from the observed responses. However, this algorithm could be manipulated by more sophisticated adversaries who can collude together and adapt their labeling strategy to target certain tasks to lower their penalty scores. In particular, the soft-penalty algorithm treats each task in isolation when assigning penalties; therefore, it is susceptible to attack by determined adversaries who can cleverly decide their responses based on honest workers' labels and the structure of the worker-task assignment graph to cause maximum "damage". For example, suppose that the subgraph of $\mathcal{B}$ between honest workers and tasks is r-right regular, that is, each task receives labels from exactly r honest workers (such graphs are commonly used in practice, see [KOS14] and [OOSY16]), and all honest workers are perfectly reliable. Now, suppose that there are $k > r$ adversaries and that each adversary provides the incorrect response to *all* tasks. Then, every task has r correct responses, all provided by honest workers, and k incorrect responses, all provided by adversaries, resulting in net penalties of $1/r$ for each honest worker and $1/k$ for each adversary (note that the degree of the workers does not affect the net penalty because the penalty received from each task is the same). Because $k > r$, adversaries receive lower penalties than do honest workers. Therefore, filtering out k workers with the highest penalties will always filter out honest workers. Furthermore, natural aggregation algorithms (such as simple

majority or weighted majority with penalties as weights) result in incorrect labels for all tasks (see Lemma 2.9). In fact, for such worst-case adversaries, we can establish the following result:

Theoretical Result (Refer to Theorem 2.10 for the formal statement). *Given any collection of honest worker responses, there exists an adversary strategy that can achieve a lower bound on the fraction of tasks whose true labels cannot be inferred correctly (better than a random guess) by* ***any*** *label aggregation algorithm that is agnostic to the worker and task identities.*

The proof relies on the fact that sophisticated adversaries can render themselves *indistinguishable* from honest workers by carefully choosing their responses. This shows that identifying such worst-case adversarial workers can be difficult. To deal with such adversarial behavior, we use the *hard-penalty* algorithm.

2.3.2 Hard-Penalty Algorithm

To deal with sophisticated adversaries, we propose a *hard* penalty allocation scheme (Algorithm 6) in which the penalty allocation for a particular task takes into account the structure of the worker-task assignment graph and the responses of other workers on *all* other tasks. In particular, instead of distributing the penalty evenly across all workers that respond to a given task, this algorithm chooses two "representative" workers to penalize for each conflict task: one each among those who provide the label $+1$ and -1. The representative workers are chosen in a load-balanced manner to "spread" the penalty across all workers and thereby avoid over-penalizing workers who provide labels for a large number of tasks. The net penalty of each worker is the sum of penalties accrued across all (conflict) tasks assigned to this worker. Intuitively, such a hard allocation of penalties will penalize workers with higher degrees (i.e. large number of assigned tasks) and many conflicts (who are potential worst-case adversaries), thereby leading to a low reputation.

To choose representative workers in a load-balanced fashion, we use the concept of *optimal semi-matchings* [HLLT03] in bipartite graphs. For a bipartite graph $G = (V_1 \cup V_2, E)$, a *semi-matching* in G is a set of edges $M \subseteq E$ such that each vertex in V_2 is incident to exactly one edge in M (note that vertices in V_1 could be incident to multiple edges in M). A semi-matching generalizes the notion of matchings on bipartite graphs. An optimal semi-matching is the semi-matching with the minimum *cost*. We use the common degree-based cost function defined as follows: for each $u \in V_1$, let $deg_M(u)$ denote the *degree* of u, that is, the number of edges in M that are incident to u, and let $cost_M(u)$ be defined as

$$cost_M(u) := \sum_{i=1}^{deg_M(u)} i = \frac{deg_M(u) \cdot (deg_M(u) + 1)}{2}$$

Then, an *optimal semi-matching* is one that minimizes $\sum_{u \in V_1} cost_M(u)$.[4] Intuitively, an optimal semi-matching *fairly* matches V_2 vertices across V_1 vertices such that the "load" on

[4]The optimal semi-matching need not be unique; for our purposes, we use any semi-matching that

any V_1 vertex is minimized. The above notion of cost is motivated by the load balancing problem for scheduling tasks on machines. Specifically, consider a set of unit-time tasks V_1 and a set of machines V_2. Suppose that each task can be processed on a subset of machines; this can be specified as a bipartite graph between V_1 and V_2. On any given machine, the tasks are executed one after the other in series. An optimal semi-matching can be thought of as an assignment of tasks to the machines such that the *flow-time*, that is, average completion time of a task, is minimized. See [HLLT03] for more details.

To determine the representative workers for each task, we compute the optimal semi-matching in the following augmented worker-task assignment graph: we split each task t_j into two copies, t_j^+ and t_j^-, and connect worker w_i to t_j^+ or t_j^- depending on whether this worker labeled the task $+1$ or -1, respectively. By definition, the optimal semi-matching yields two representative workers for each task t_j—one connected to t_j^+ and the other connected to t_j^-. As in the soft-penalty algorithm, we only consider conflict tasks when creating this augmented bipartite graph. The worker degrees in the optimal semi-matching then constitute their net penalties. Algorithm 6 describes the hard-penalty algorithm.

2.3.3 Connection between soft- and hard-penalty algorithms

Although the soft- and hard-penalty algorithms appear different on surface, the former can be interpreted as a random, normalized variant of the latter. Specifically, suppose we choose a random semi-matching M in the augmented worker-task assignment graph $\mathcal{B}^{cs}$, defined in Algorithm 6, and assign the penalty $deg_M(w_i)/deg_{\mathcal{B}^{cs}}(w_i)$ to worker w_i, where $deg_{\mathcal{B}^{cs}}(w_i)$ is the degree of worker w_i in $\mathcal{B}^{cs}$. When the random semi-matching is constructed by mapping each copy t_j^+ (or t_j^-) of task t_j uniformly at random to a worker connected to it, the probability that it will be mapped to worker $w_i \in W_j$ is equal to $1/deg_{\mathcal{B}^{cs}}(t_j^+)$ (or $1/deg_{\mathcal{B}^{cs}}(t_j^-)$), or equivalently, $1/d_j^+$ (or $1/d_j^-$). Therefore, the expected degree $\mathbb{E}[deg_M(w_i)]$ of worker w_i is equal to $\sum_{t_j \in \mathcal{T}_i \cap \mathcal{T}_{cs}} s_{ij}$, where $s_{ij} = 1/d_j^+$ if $\mathcal{L}_{ij} = +1$ and $1/d_j^-$ if $\mathcal{L}_{ij} = -1$. Because the degree $deg_{\mathcal{B}^{cs}}(w_i)$ of worker w_i is equal to $|\mathcal{T}_i \cap \mathcal{T}_{cs}|$, it follows that the expected penalty of worker w_i is equal to $\mathbb{E}[deg_M(w_i)]/deg_{\mathcal{B}^{cs}}(w_i) = \sum_{t_j \in \mathcal{T}_i \cap \mathcal{T}_{cs}} s_{ij} / |\mathcal{T}_i \cap \mathcal{T}_{cs}|$, which is exactly the penalty allocated by the soft-penalty algorithm. It thus follows that the expected penalties under the above random, normalized variant of the hard-penalty algorithm are equal to the penalties allocated by the soft-penalty algorithm. When all workers are assigned the same number of tasks, the expected penalty assigned by the random hard-penalty algorithm is equal to the penalty assigned by the soft-penalty algorithm, but scaled by a constant factor.

With the above interpretation of the soft-penalty algorithm, it follows that the hard-penalty algorithm differs from the soft-penalty algorithm in two key aspects: it (a) does not normalize the penalties by degrees and (b) uses optimal semi-matchings as opposed to random semi-matchings. The absence of degree-based normalization of the penalties results in significant penalization of high-degree workers. The use of the optimal semi-matching results in a more balanced allocation of penalties by optimizing a global objective function. Both

minimizes the cost. In Section 2.4.2, we show that the worst-case performance of the hard penalty algorithm is agnostic to the choice of the optimal semi-matching.

of these effects make the hard-penalty algorithm conservative and robust to sophisticated adversary strategies, as established theoretically in Section 2.4. The above connection also suggests that the random, normalized variant of the hard-penalty algorithm should have performance similar to that of the soft-penalty algorithm. We explore this aspect theoretically at the end of Section 2.4.1.

2.4 Theoretical Results

Our reputation algorithms are analytically tractable, and we establish their theoretical properties below. Our analysis is aimed at deriving the conditions under which our algorithms separate adversaries from honest workers. We use uppercase boldface letters (say, $\boldsymbol{X}$) to denote random variables, unless it is clear from the context. The proofs of all results are given in Appendix B.

2.4.1 Soft-penalty algorithm: common adversary strategies

First, we analyze the performance of the soft-penalty algorithm. We assume that honest workers adopt the one-coin model, where each worker w is characterized by parameter $\mu_w \in [0, 1]$ that specifies the probability that w provides the correct response for any task. We choose this model because it is the simplest non-trivial model to analyze, and it has been widely studied in existing literature [GKM11, DDKR13, KOS14, ZCZJ14, GZ16]. We focus on two settings: (a) the classical setting in which there are no adversaries ($A = \emptyset$) and (b) the setting in which adversaries adopt the Uniform strategy.

Definition 2.1 (Uniform strategy)**.** Every adversarial worker provides the same $+1$ response to all tasks assigned to it.

We consider the Uniform strategy because of its ubiquity and simplicity. It is commonly observed in practice [VdVE11], and our analysis of real-world datasets (discussed in detail in Section 2.5) reveals that a significant fraction of workers ($\approx 30\%, 17\%$, and 9% of workers in the `stage2, task2`, and `temp` datasets, respectively) provide uniform labels for all assigned tasks. Note that it is not captured by the standard one-coin model, in which workers assign the label y_j or $-y_j$ to task t_j, where y_j is the true label. It can be adopted by both "lazy" and "smart" adversaries. Lazy workers adopt this strategy to maximize the number of tasks they label and the corresponding payment they obtain. Smart adversaries adopt this strategy if it is known a priori that the prevalence (or proportion) γ of tasks with true labels $+1$ is large, say, above 90%. For instance, medical images showing tumors contain a large proportion of benign ones and a correspondingly small proportion of malignant ones, leading a "smart" worker to label all assigned images as benign without carefully considering each image. Therefore, the Uniform strategy actually comprises a spectrum of strategies of varying degrees of "smartness", with higher values of γ indicating smarter strategies.

A natural way to identify adversaries who adopt the Uniform strategy is to classify all workers who have labeled *all* assigned tasks $+1$ as adversaries. However, we show that this

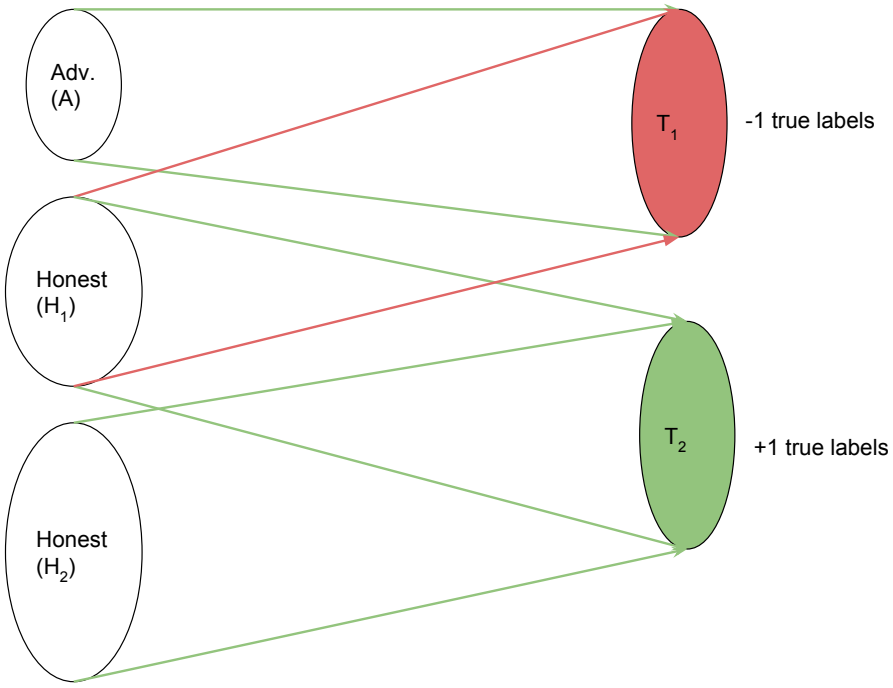

Figure 2.1: Example where filtering workers who provide only +1 labels performs poorly

approach can have arbitrarily large *misclassification rate*, because it may misclassify almost all (asymptotic fraction approaching 1) perfectly reliable honest workers as adversarial.

Lemma 2.1 (Hardness of identifying Uniform adversarial workers)**.** *Consider a simple binary classifier* $\hat{\boldsymbol{I}}_{\text{natural}}(\cdot)$ *as follows:*

$$\hat{\boldsymbol{I}}_{\text{natural}}(w_i) = \begin{cases} \textit{adversarial}, & \textit{if } w_i \textit{ gives all } +1 \textit{ labels} \\ \textit{honest}, & \textit{o.w.} \end{cases}$$

Then, the misclassification rate, that is, fraction of incorrectly classified workers, of the above classifier can be arbitrarily close to 1.

Proof. Suppose the collection of tasks partitions into two sets, T_1 and T_2, consisting of tasks that have true labels -1 and $+1$, respectively (see Figure 2.1). The adversary A follows the Uniform strategy and labels all tasks in T_1 as $+1$. There are two groups of honest workers—H_1 and H_2—who are perfectly reliable, that is, they always provide the correct label. The workers in H_1 label all tasks in $T_1 \cup T_2$, and workers in H_2 label only the tasks in T_2. Now, because honest workers H_2 give only $+1$ labels, the classifier $\hat{\boldsymbol{I}}_{\text{natural}}$ misclassifies all honest workers in H_2 as adversaries. In other words, we have

$$\frac{1}{|H_1 \cup H_2|} \sum_{h \in H_1 \cup H_2} \mathbf{1}\left[\hat{\boldsymbol{I}}_{\text{natural}}(h) \neq \textit{honest}\right] = \frac{|H_2|}{|H_1 \cup H_2|}$$

where $\mathbf{1}[\cdot]$ denotes the indicator function. Suppose H_2 comprises a large fraction of honest workers, that is, $|H_2| = (1-\rho)\,|H_1 \cup H_2|$ for some "small" $\rho > 0$. Then, it follows from the above equation that the misclassification rate of the natural classifier $\hat{\boldsymbol{I}}_{\text{natural}}(\cdot)$ is $1-\rho$, which can be arbitrarily close to 1. □

The above result shows that the problem of identifying adversaries is nontrivial even for the Uniform strategy case, and our analysis below provides precise conditions under which we can identify such adversarial workers. In particular, for the scenario outlined in the proof above, our soft-penalty algorithm assigns a penalty of $\frac{1}{|A|}$ to adversaries, $\frac{1}{|H_1|}$ to honest workers in H_1, and zero penalty to honest workers in H_2 (because they are not part of any conflicts). Consequently, as long as $|H_1| > |A|$, our algorithm correctly separates out honest workers from adversaries by choosing a threshold $\theta \in \left(\frac{1}{|H_1|}, \frac{1}{|A|}\right)$ and classifying all workers with penalty $> \theta$ as adversarial.

Because the performance of the algorithm depends on the specific crowdsourced classification instance (worker-task assignment graph, true task labels, reliabilities of honest workers), we conduct a probabilistic analysis under a natural generative model. For our analysis, we focus on worker-task assignment graphs $\mathcal{B}$ that are (l, r)-regular, in which each worker is assigned l tasks and each task is labeled by r workers. These assignment graphs are analytically tractable and have been shown in [KOS14] to achieve order-optimal performance (with respect to the best combination of task assignment and inference algorithm) when given a certain budget for task assignment. To generate the crowdsourcing instance, we use the probabilistic model of crowdsourced labeling proposed in [KOS14] but extended to incorporate adversarial workers:

Generative model. Suppose the fraction $q \in (0, 1]$ of honest workers, number of tasks m, number of workers n, worker degree $l > 1$, and task degree $r > 1$ are fixed. Let $\gamma \in [0, 1]$ denote the prevalence (or proportion) of tasks with true labels $+1$ and $F(\cdot)$, the cumulative distribution function (CDF) of the honest worker reliabilities under the one-coin model, with $\mu \in [0, 1]$ denoting the mean. Sample a crowdsourced classification instance as follows:

1. *Worker-task assignment graph:* Assign m tasks to n workers using the *configuration model*—take $n \cdot l$ half-edges for worker nodes and $m \cdot r$ half-edges for the task nodes, pick a random permutation of worker half-edges, and map them to task half-edges.

2. *True task labels:* For each task t_j, sample the true label $\boldsymbol{Y}_j \in \{-1, +1\}$ independently according to the Bernoulli distribution with $\mathbb{P}[\boldsymbol{Y}_j = +1] = \gamma$.

3. *Worker identities:* For each worker w_i, set its identity to *honest* with probability q and *adversarial* with probability $1 - q$.

4. *Honest worker reliabilities and responses:* If w_i is honest, sample its reliability $\boldsymbol{M}_i = \mu_i$ from the distribution $F(\cdot)$. For each task t_j assigned to w_i, set the response $w_i(t_j)$ to $\boldsymbol{Y}_j$ with probability μ_i and $-\boldsymbol{Y}_j$ with probability $1 - \mu_i$.

5. *Adversarial worker responses:* If w_i is adversarial, set the response $w_i(t_j) = +1$ for all tasks t_j assigned to w_i.

The above generative model may be justified as follows. First, the *configuration model* is a simple random construction to generate graphs that is popular in random graph literature [Bol01]. It may result in a graph with multi-edges (where two nodes are connected by more than one edge); however, the number of double-edges converges to a Poisson distribution with mean $(l-1)(r-1)/2$ (see [Bol01]). Therefore, the proportion of nodes with multi-edges is $\approx lr/n$, which tends to zero as $n \to \infty$ as long as $l = o(n)$ and r is constant.

The model for true task labels, worker identities, and reliabilities may be justified by supposing that the m tasks $\mathcal{T} = \{t_1, \ldots, t_m\}$ are drawn from a "large" population of tasks with a prevalence γ of $+1$ tasks and workers are drawn from a "large" population with a proportion $1-q$ of adversaries and a proportion q of honest workers, whose reliabilities have the distribution $F(\cdot)$. Then, the distributional assumptions for the true task labels, worker identities, and honest worker reliabilities will be met when the task assignment is randomized and there is no distinction between sampling with and without replacement because of the large population sizes. The model for generating honest worker responses is the standard one-coin model. See [KOS14] for a detailed discussion of the settings under which the above probabilistic model is reasonable.

For our theoretical analysis, we assume that non-conflict tasks (with all $+1$ or -1 labels) are *not* ignored/dropped for the purposes of penalty computation. This assumption makes the analysis less cumbersome and may be justified by noting that for a large enough r, the probability that a task will be non-conflict is low. Even if a task is non-conflict, we expect little impact from its inclusion because the penalty from this task will be $1/r$, which is negligible for large values of r. We also tested this assertion numerically and observed negligible differences in the performances of the two variants (with and without dropping high-degree non-conflict tasks) of the soft-penalty algorithm (see Section 2.5).

Analysis of expected penalties. We first analyze the expected penalties received by honest and adversarial workers under the abovementioned generative model and identify the conditions for population parameters q, μ, and γ under which honest workers receive lower expected penalties. Let $\mathbf{PEN}_i$ denote the penalty assigned by the soft-penalty algorithm to worker w_i; note that it is a random variable under the abovementioned generative model.

First, we focus on the classical setting in which there are no adversarial workers; therefore, $A = \emptyset$. We obtain the following result:

Theorem 2.2 (Reputations consistent with reliabilities)**.** *When $q = 1$ (i.e., there are no adversarial workers) and $\mu > \frac{1}{2}$, we have*

$$\mathbb{E}\left[\mathbf{PEN}_i \mid \boldsymbol{M}_i = \mu_1\right] < \mathbb{E}\left[\mathbf{PEN}_i \mid \boldsymbol{M}_i = \mu_2\right] \iff \mu_1 > \mu_2$$

for any worker w_i.

Theorem 2.2 shows that the expected reputation scores are consistent with honest workers' reliabilities: as the honest worker's reliability decreases, the expected penalty increases, that

is, the expected reputation score decreases. As honest worker reliabilities capture their propensities to make mistakes, our algorithm flags workers who are prone to making mistakes, as desired. Consequently, filtering out low-reputation workers filters out workers with low reliabilities (we make this claim precise in Theorem 2.4 below).

Next, we consider the case in which $A \neq \emptyset$ and there is a fraction $1-q$ of workers who are adversarial and adopt the Uniform strategy. Let p_h and p_a denote the expected penalties that a worker receives conditioned on being honest and adversarial, respectively, that is,

$$p_h = \mathbb{E}\left[\mathbf{PEN}_i \mid w_i \text{ is honest }\right] \quad \text{and} \quad p_a = \mathbb{E}\left[\mathbf{PEN}_i \mid w_i \text{ is adversarial }\right]$$

Because of symmetry, these expectations do not depend on worker indices. Then, we obtain the following result:

Theorem 2.3 (Penalties under Uniform strategy)**.** *When $q < 1$, $\mu > \frac{1}{2}$, and the adversaries adopt the* Uniform *strategy, we obtain*

$$p_h < p_a \iff \left(q\mu > \frac{1}{2} \quad \text{and} \quad \frac{\mu}{1-\mu} \cdot h(\mu, q) > \frac{\gamma}{1-\gamma}\right)$$

where $h(\mu, q)$ is a strictly increasing function in q for a fixed μ, and it is defined as

$$h(\mu, q) = \frac{g(1-Q) - g(Q)}{g(P) - g(1-P)} \text{ where } P := q\mu + (1-q), Q := 1 - q\mu, \text{ and } g(x) := \frac{1-x^r}{r \cdot (1-x)}.$$

The above result reveals the conditions for the parameters q, μ, and γ under which the soft-penalty algorithm is successful in assigning lower penalties to honest workers than to adversaries.

To understand this, we first focus on the condition $q\mu > 1/2$. Note that because $\mu \leq 1$, this condition implies that the population must consist of more honest workers than adversaries ($q > 1/2$). This is because our algorithm is designed to identify "outlier" response patterns—those which deviate from the majority—and for adversaries to be declared outliers, they must necessarily be in the minority.

Furthermore, note that the necessary condition $\mu > 1/(2q)$ implies that for our algorithm to be successful, the average reliability μ of the honest workers must be "large enough"; specifically, it must exceed $\frac{1}{2q}$ (for a fixed q). To obtain an intuitive understanding of this condition, note the following. Consider a task with true label $+1$. Then, in expectation, there are $rq\mu$ honest workers and $(1-q) \cdot r$ adversaries who provide the response $+1$ and $rq \cdot (1-\mu)$ honest workers who provide the response -1. Now, the adversaries will agree with the majority if and only if $r \cdot (q\mu + (1-q)) \geq rq \cdot (1-\mu)$, that is, $\mu \geq 1 - 1/(2q)$. Similarly, when the true label of a task is -1, then in expectation, there are $r \cdot (q \cdot (1-\mu) + (1-q))$ workers providing $+1$ label and $rq\mu$ workers providing -1 label. Again, the adversaries will be in the majority in expectation if and only if $\mu \leq 1/(2q)$. It thus follows that if $\mu \in [1 - \frac{1}{2q}, \frac{1}{2q}]$, then the adversaries are always in majority in expectation, and therefore, they will receive a

lower penalty. Because $\mu > 1/2$ and $1 - \frac{1}{2q} \leq 1/2$, a necessary condition for the worker to receive a lower penalty when being honest is therefore $\mu > \frac{1}{2q}$, that is, $q\mu > \frac{1}{2}$.

Now assuming that the first condition is met, that is, $q\mu > 1/2$, we focus on the second condition: $\frac{\mu}{1-\mu} \cdot h(\mu, q) > \frac{\gamma}{1-\gamma}$. When $\gamma = 1$, this condition is not met (unless $\mu = 1$), and therefore, honest workers receive higher expected penalties than adversaries. This is because if all tasks have a true label $+1$, then in expectation, there is a fraction $q\mu + 1 - q > 1/2$ (because $q\mu > 1/2$) of workers providing the label $+1$ to each task, implying that the adversaries always agree with the majority label for each task. As a result, our algorithm filters out honest workers; however, it must be noted that adversaries actually have higher (perfect) reliabilities in this special case. Similarly, when $\mu = 1$, that is, honest workers are perfectly reliable, the condition is always met because honest workers are in the majority at each task (in expectation); specifically, when the true label is $+1$, all responses are $+1$, and when the true label is -1, all honest workers (a fraction $q > 1/2$) provide the label -1.

Next, we investigate the performance of our algorithm as the adversary strategies become "smarter". As noted above, the Uniform strategy comprises a spectrum of strategies of varying degrees of "smartness", with higher values of γ indicating smarter strategies.

Corollary 2.3.1 (Penalties under smarter adversary strategies)**.** *Suppose $\mu > \frac{1}{2}$ is fixed and $q\mu > \frac{1}{2}$. Then, it follows that*

$$p_h < p_a \iff q > h_\mu^{-1}\left(\frac{\gamma}{1-\gamma}\right)$$

where $h_\mu(q) := \frac{\mu}{1-\mu} \cdot h(\mu, q)$ and $h_\mu^{-1}(\cdot)$ is the inverse of $h_\mu(\cdot)$. In other words, for fixed $\mu > \frac{1}{2}$, we require a minimum fraction of honest workers to ensure that adversaries receive a higher penalty than honest workers, and this fraction increases as γ increases.

The above result shows that as the adversary strategies become smarter, it becomes more difficult to distinguish them from honest workers. Specifically, because $h_\mu^{-1}(\cdot)$ is a strictly increasing function, we require honest workers to have a larger majority as γ increases to ensure that they receive lower expected penalties than adversaries.

Asymptotic identification of adversaries and honest workers. Assuming that the expected penalties of adversaries and honest workers are separated, we now derive the asymptotic error rates, defined as the expected fraction of errors, of the soft-penalty algorithm as $n \to \infty$ when (1) there are no adversaries and (2) adversaries adopt the Uniform strategy.

To analyze the error rates, we consider the following threshold-classifier $\hat{\boldsymbol{I}}_\theta(\cdot)$ based on the soft-penalty algorithm: given a penalty threshold $\theta \in \mathbb{R}$, define the binary classifier

$$\hat{\boldsymbol{I}}_\theta(w_i) = \begin{cases} \text{honest}, & \text{if } \mathbf{PEN}_i \leq \theta \\ \text{adversarial}, & \text{o.w.} \end{cases}$$

Let $\boldsymbol{I}(w_i)$ denote the latent true identity of worker w_i. Note that both $\boldsymbol{I}(w_i)$ and $\hat{\boldsymbol{I}}_\theta(w_i)$ are random variables under our generative model.

Case 1: no adversaries. We first consider the case in which there are no adversaries, that is, $q = 1$, so that $\boldsymbol{I}(w_i) = \textit{honest}$ for all workers w_i. In this case, Theorem 2.2 shows that workers with higher reliabilities receive lower expected penalties. Furthermore, we now show that the threshold classifier correctly classifies high-reliability workers as honest with high probability as $n \to \infty$:

Theorem 2.4 (Identification of high-reliability workers)**.** *Suppose $q = 1$ and $\mu > \frac{1}{2}$. Given $\theta \in (0, 1)$ and $\varepsilon \in (0, \frac{1}{\sqrt{2}})$ such that $\theta - \varepsilon \in (g(1-\mu), g(\mu))$, define $\hat{\mu}(\theta) := \frac{g(\mu)+\varepsilon-\theta}{g(\mu)-g(1-\mu)}$, where the function $g(\cdot)$ was defined in Theorem 2.3. Then, under the generative model, we obtain*

$$\frac{1}{n}\sum_{i=1}^{n} \mathbb{P}\left(\hat{\boldsymbol{I}}_\theta(w_i) \neq \boldsymbol{I}(w_i) \text{ and } \boldsymbol{M}_i > \hat{\mu}(\theta)\right) \leq \frac{l^2 r^2}{n-1} + \exp\left(-\frac{2l\varepsilon^2}{(1-1/r)^2}\right)$$

When $l = \log n$ and r is fixed, we obtain

$$\frac{1}{n}\sum_{i=1}^{n} \mathbb{P}\left(\hat{\boldsymbol{I}}_\theta(w_i) \neq \boldsymbol{I}(w_i) \text{ and } \boldsymbol{M}_i > \hat{\mu}(\theta)\right) = O\left(\frac{1}{n^{2\varepsilon^2}}\right) \text{ as } n \to \infty$$

Theorem 2.4 provides an upper bound for the *error rate* or *misclassification rate* of the threshold classifier. We say that the classifier makes an *error* if it classifies a worker whose reliability is higher than $\hat{\mu}(\theta)$ as adversarial, and the misclassification rate is defined as the expected fraction of errors. As $n \to \infty$, the first term, $l^2 r^2/(n-1)$, in the error bound goes to 0 as long as $l = o(\sqrt{n})$. With the task degree r fixed, the second term goes to zero as long as $l \to \infty$ when $n \to \infty$. Upon combining these two observations, it follows that taking $l = \log n$ yields the error bound of $O(1/n^{2\varepsilon^2})$ for a fixed $\varepsilon \in (0, \frac{1}{\sqrt{2}})$ and r, as $n \to \infty$. In other words, our result shows that as long as we collect $\log n$ labels from each worker and a fixed number of labels for each task, we will classify workers with reliabilities higher than $\hat{\mu}(\theta)$ as honest with a high probability as $n \to \infty$. Although the number of labels collected from each worker must tend to infinity, it must only grow logarithmically in the total number of workers n. Finally, if the population reliability μ is known, we can determine the value of threshold θ for a given reliability threshold $\hat{\mu}(\theta)$.

The proof of Theorem 2.4 (given in Appendix B.2.3) relies on establishing that the worker penalties are concentrated around their respective expectations, for which we need the worker-task assignment graph $\mathcal{B}$ to be locally tree-like:

Definition 2.2 (Locally tree-like assignment graphs)**.** An (l, r)-regular worker-task assignment graph $\mathcal{B}$ is said to be *D-locally tree-like* at a worker node w_i if the subgraph $\mathcal{B}_{w_i,D}$, consisting of nodes at a distance of at most D from w_i, is a tree.

For our purposes, it suffices to have $\mathcal{B}_{w_i,2}$ be a tree. Note that the subgraph $\mathcal{B}_{w_i,2}$ consists of the worker node w_i; tasks labeled by w_i, that is, the set $\mathcal{T}_i$; and workers $\bigcup_{j\in\mathcal{T}_i} W_j$ who labeled the tasks in $\mathcal{T}_i$. [KOS14] shows that a random construction of the assignment graph using the configuration model ensures that $\mathcal{B}_{w,2}$ is a tree with a high probability as $n \to \infty$ for a randomly chosen worker w.

Lemma 2.5 (Random construction ensures local tree-structure)**.** *If $\mathcal{B}$ is random (l, r)-regular constructed according to the configuration model, then for a randomly chosen worker $\boldsymbol{w}$*

$$\mathbb{P}\left(\mathcal{B}_{\boldsymbol{w},2} \text{ is not a tree}\right) \leq \frac{l^2 r^2}{n-1}$$

The proof of the above result is in Appendix B.2.1 and follows the arguments in [KOS14]. Based on the result of Lemma 2.5, the proof of Theorem 2.4 proceeds in two steps. First, whenever the configuration model generates an assignment graph $\mathcal{B}$ that is not locally tree-like, we immediately declare an error, incurring an error probability that is bounded above by $l^2r^2/(n-1)$; this yields the first term in the error bound. Note that this error bound increases as r—the number of labels for each task—increases, since the probability that there is a cycle in $\mathcal{B}_{\boldsymbol{w},2}$, and therefore it is not a tree, increases as we add more edges to $\mathcal{B}$. Second, when $\mathcal{B}_{w_i,2}$ is indeed a tree, we obtain the second term in the error bound by invoking the following concentration result:

Lemma 2.6 (Concentration of honest worker penalties)**.** *Suppose that $q = 1$ and $\mathcal{B}_{w_i,2}$ is a tree. Under the generative model and for a fixed reliability $\boldsymbol{M}_i = \mu_i$, given any $\varepsilon > 0$, the penalty assigned to honest worker w_i concentrates as*

$$\mathbb{P}\left(\mathbf{PEN}_i \geq \mathbb{E}[\mathbf{PEN}_i \mid \boldsymbol{M}_i = \mu_i] + \varepsilon \,\middle|\, \boldsymbol{M}_i = \mu_i\right) \leq \exp\left(\frac{-2l\epsilon^2}{(1-1/r)^2}\right)$$

The above result holds for *any* fixed value of the reliability μ_i. The proof (given in Appendix B.2.2) relies on expressing the penalty scores as an average of l random variables and then invoking Hoeffding's concentration bound. The local tree-like property of the assignment graph $\mathcal{B}$ at worker node w_i ensures that the l random variables are mutually independent (which is required for Hoeffding's inequality).

Case 2: adversaries adopt the Uniform strategy. Next, we consider the case in which there is a fraction $1-q$ of workers who are adversaries and adopt the Uniform strategy. Theorem 2.3 above provides the necessary and sufficient conditions for an honest worker to receive a lower expected penalty than an adversary, that is, for $p_h < p_a$. Under these conditions, we obtain the following result:

Theorem 2.7 (Identification of honest and adversarial workers)**.** *Suppose $p_h < p_a$ and let $\theta \in (p_h + \varepsilon, p_a - \varepsilon)$ for some ε small enough such that $0 < \epsilon < (p_a - p_h)/2$. Then, under the generative model we obtain*

$$\frac{1}{n}\sum_{i=1}^{n} \mathbb{P}\left(\hat{\boldsymbol{I}}_\theta(w_i) \neq \boldsymbol{I}(w_i) \,\middle|\, \boldsymbol{I}(w_i) = \textit{honest}\right) \leq \frac{l^2r^2}{n-1} + \exp\left(-\frac{2l\varepsilon^2}{(1-1/r)^2}\right) + F\left(\hat{\mu}(q,\theta)\right)$$

$$\frac{1}{n}\sum_{i=1}^{n} \mathbb{P}\left(\hat{\boldsymbol{I}}_\theta(w_i) \neq \boldsymbol{I}(w_i) \,\middle|\, \boldsymbol{I}(w_i) = \textit{adversarial}\right) \leq \frac{l^2r^2}{n-1} + \exp\left(-\frac{2l\varepsilon^2}{(1-1/r)^2}\right)$$

where $\hat{\mu}(q,\theta)$ is such that $\hat{\mu}(1,\theta) = \hat{\mu}(\theta)$ and $\hat{\mu}(q,\theta) < \mu$ for all $q \in (0,1]$ and $\theta \in (p_h+\varepsilon, p_a-\varepsilon)$.

When $l = \log n$ and r is fixed, we obtain

$$\frac{1}{n}\sum_{i=1}^{n}\mathbb{P}\left(\hat{\boldsymbol{I}}_\theta(w_i) \neq \boldsymbol{I}(w_i) \,\middle|\, \boldsymbol{I}(w_i) = \textit{honest}\right) = O\left(\frac{1}{n^{2\varepsilon^2}}\right) + F\left(\hat{\mu}(q,\theta)\right)$$
$$\frac{1}{n}\sum_{i=1}^{n}\mathbb{P}\left(\hat{\boldsymbol{I}}_\theta(w_i) \neq \boldsymbol{I}(w_i) \,\middle|\, \boldsymbol{I}(w_i) = \textit{adversarial}\right) = O\left(\frac{1}{n^{2\varepsilon^2}}\right)$$

The precise expression for $\hat{\mu}(q,\theta)$ is involved and is given in Appendix B.2.4. Theorem 2.7 provides the misclassification rate of our algorithm when the population parameters q, μ, and γ satisfy the conditions of Theorem 2.3, ensuring that honest workers receive a lower expected penalty than adversaries. Following the arguments from the discussion under Theorem 2.4 above, it can be seen that when $l = \log n$ and r is fixed, the fraction of adversaries that is misclassified is $O(1/n^{2\varepsilon^2})$. On the other hand, the fraction of honest workers that is misclassified scales as $O(1/n^{2\varepsilon^2}) + F(\hat{\mu}(q,\theta))$. The first term tends to zero as $n \to \infty$. The second term denotes the probability that honest worker reliability is less than or equal to $\hat{\mu}(q,\theta)$. In other words, our algorithm misclassifies low-reliability workers as adversaries. In the special case when all honest workers have the same reliability μ, it immediately follows that the probability density function is a point mass at μ, from which it follows that $F(\hat{\mu}(q,\theta)) = 0$ because $\hat{\mu}(q,\theta) < \mu$. In this case, the misclassification error for the honest workers also tends to zero as $n \to \infty$.

We note that the dependence of the honest worker misclassification rate on $F\left(\hat{\mu}(q,\theta)\right)$ is fundamental to our algorithm. As an example, consider the case when the reliability distribution is a two-point distribution with probability mass μ at 1 and the remaining $1-\mu$ mass at 0. This distribution results in two types of honest workers: workers who always provide the correct response and those that always provide the incorrect response. Note that the average reliability under this distribution is μ. Let p_0 and p_1 denote the expected penalties under our generative model for a worker conditioned on being honest with reliabilities 0 and 1, respectively. Then, it follows that

$$\begin{aligned} p_1 &= \gamma \cdot g(1-P) + (1-\gamma)\cdot g(Q) \\ p_a &= \gamma \cdot g(1-P) + (1-\gamma)\cdot g(1-Q) \\ p_0 &= \gamma \cdot g(P) + (1-\gamma)\cdot g(1-Q) \end{aligned}$$

From Theorem 2.3, it follows that $P > 1/2$ and $Q < 1/2$ is necessary to ensure $p_h < p_a$. Combined with the fact that $g(\cdot)$ is increasing, this implies that $g(Q) < g(1-Q)$ and $g(1-P) < g(P)$. As a result, we obtain $p_1 < p_a < p_0$. It now follows that when $n \to \infty$, the penalties of honest workers with reliability 0 concentrate around p_0, and consequently, they are classified as adversarial whenever the threshold $\theta < p_a$, resulting in a misclassification error of $1-\mu$. However, it should be noted that honest workers classified as adversaries indeed have low reliabilities.

Similar to the proof of Theorem 2.4 above, the proof of Theorem 2.7 proceeds in two steps. The first term in the error bound of Theorem 2.7 comes from Lemma 2.5 because we

immediately declare an error whenever the assignment graph is not locally tree-like. The second term comes from the case when $\mathcal{B}_{w_i,2}$ is indeed a tree by invoking the following concentration result:

Lemma 2.8 (Concentration of worker penalties)**.** *Suppose that $q < 1$ and $\mathcal{B}_{w_i,2}$ is a tree. Under the generative model, given any reliability value $\hat{\mu} \in (0,1)$ and $\varepsilon > 0$, the penalty assigned to worker w_i concentrates as*

$$\mathbb{P}\left(\mathbf{PEN}_i \geq \mathbb{E}[\mathbf{PEN}_i \mid \boldsymbol{M}_i = \hat{\mu}] + \varepsilon \;\middle|\; \boldsymbol{I}(w_i) = \textit{honest}\right) \leq \exp\left(\frac{-2l\epsilon^2}{(1-1/r)^2}\right) + F(\hat{\mu})$$

and

$$\mathbb{P}\left(\mathbf{PEN}_i \leq p_a - \varepsilon \;\middle|\; \boldsymbol{I}(w_i) = \textit{adversarial}\right) \leq \exp\left(\frac{-2l\epsilon^2}{(1-1/r)^2}\right)$$

The proof of the above result is similar to that of Lemma 2.6. For the case of adversarial workers, we use Hoeffding's argument to establish the concentration. For the case of honest workers, the first term follows directly from Lemma 2.6 when w_i has reliability $\mu_i > \hat{\mu}$ and the second term is the probability that the reliability $\mu_i \leq \hat{\mu}$.

The above results establish that the soft-penalty algorithm successfully identifies low-reliability honest workers and adversaries adopting the Uniform strategy asymptotically with high probability. Note that all results also extend to the random, normalized variant of the hard-penalty algorithm mentioned in Section 2.3.3, where the expectation is taken over the generative model *and* the randomized hard-penalty algorithm—see Appendix B.2.6 for the details.

We would like to end our discussion by pointing out that similar results can be derived if it is assumed that honest workers employ the two-coin model (instead of the one-coin model assumed in the preceding analysis), where each worker is characterized by two parameters (α_w, β_w). However, the precise error bounds require significantly more notation to explain, without adding too much in terms of insights. Instead, we evaluate the performance of our reputation algorithms for the two-coin model in the numerical experiments and show that it is still able to identify uniform adversaries and low-reliability honest workers (see Section 2.5.1).

2.4.2 Hard-penalty algorithm: sophisticated adversary strategies

In the preceding analysis, we focused on common adversary strategies in which adversaries were not intentionally malicious. However, existing studies provide ample evidence for the presence of workers with malicious intent in public crowdsourcing systems, where workers choose which tasks to label and the worker labels are public. These workers are usually hired online by an attacker [WWZ+12] to create fake accounts and manipulate ratings/reviews to alter the aggregate ratings or rankings received by tasks. Specific examples include workers on Digg altering the "popularity" of advertisements and phishing articles [TMLS09], fake review groups collaboratively targeting products on Amazon [MLG12], workers providing fake ratings

and reviews to alter the aggregate ratings of restaurants on Yelp [MKKSM13], and malicious crowd behavior in online surveys [GKDD15]. Refer to recent work of [WWZZ14] for more examples. Motivated by these examples, we study settings with *sophisticated adversaries*, who are defined as follows:

Definition 2.3 (Sophisticated adversaries)**.** Sophisticated adversaries provide labels with the objective of maximizing the number of tasks whose inferred labels are different from the labels the tasks would otherwise have received from a standard label aggregation algorithm. They are computationally unbounded and colluding, and they possess knowledge about the labels provided by honest workers; therefore, they can adopt arbitrary labeling strategies.

Our definition allows sophisticated adversaries to not just be malicious but also be capable of executing the most complex strategies. In practice, an adversary may adopt feasible strategies with varying complexities depending on the application context and their objectives. Focusing on the most sophisticated adversary makes our analysis broadly applicable, independent of the application context. Further, we do not restrict the structure of the worker-task assignment graph because unlike in a crowdsourced-classification task, we have no control on which tasks each worker labels.

We first note that existing label aggregation algorithms can have arbitrarily bad performance in the presence of sophisticated adversaries, even if we assume that all honest workers are perfectly reliable:

Lemma 2.9 (Hardness of recovering true task labels)**.** *Suppose honest workers are perfectly reliable, that is, they always provide the correct response. Let the assignment graph between honest workers and tasks be such that each task receives responses from at most $r > 1$ honest workers. Suppose there are $k > r$ sophisticated adversaries, and each adversary provides the incorrect label on all tasks. Then, both the simple majority and EM (initialized by majority estimates) label aggregation algorithms output the incorrect label for all tasks.*

The above result characterizes the performance of label aggregation algorithms which are designed to infer true task labels, as opposed to identifying adversarial workers. Because sophisticated adversaries aim to maximize the number of tasks whose inferred labels are incorrect, the lemma shows that for standard label aggregation algorithms like simple majority and EM, the adversaries can cause arbitrarily bad "damage". Consequently, in our theoretical analysis below, we focus on the accuracy of label aggregation algorithms in the presence of sophisticated adversaries, as opposed to the misclassification rate, as was done in Section 2.4.1.

Lemma 2.9 holds for r-right regular graphs, that is, each task receives exactly r honest worker labels, which are commonly used in practice (see the discussion in Section 2.4.1). Therefore, standard label aggregation algorithms can have very poor accuracy in recovering true task labels in the presence of sophisticated adversaries. In fact, we can actually establish something much stronger—in the presence of sophisticated adversaries, there exists a lower bound on the number of tasks whose true label cannot be inferred correctly (better than random guess) *irrespective* of the label aggregation algorithm used to aggregate the worker responses.

Lower bound on number of tasks that receive incorrect labels. To state our result, we need the following additional notation. We represent any label aggregation algorithm as a *decision rule* $\boldsymbol{R} : \mathcal{L} \to \{-1, +1\}^m$, which maps the observed response matrix $\mathcal{L}$ to a set of output labels for each task. Because of the absence of any auxiliary information about the workers or the tasks, the class of decision rules, say $\mathcal{C}$, is invariant to permutations of the identities of workers and/or tasks. More precisely, $\mathcal{C}$ denotes the class of decision rules that satisfy $\boldsymbol{R}(\mathcal{P} \cdot \mathcal{L} \cdot \mathcal{Q}) = \boldsymbol{R}(\mathcal{L}) \cdot \mathcal{Q}$ for any $n \times n$ permutation matrix $\mathcal{P}$ and $m \times m$ permutation matrix $\mathcal{Q}$. We say that a task is *affected* if a decision rule outputs the incorrect label for the task, and we define the *quality* of a decision rule $\boldsymbol{R}(\cdot)$ as the *worst-case* number of affected tasks over all possible true labelings of the tasks and adversary strategies given a *fixed* set of honest worker responses. Fixing the responses provided by honest workers allows isolation of the effect of the adversary strategy on the accuracy of the decision rule. Considering the worst-case over all possible true task labelings makes the quality metric robust to ground-truth assignments, which are typically application specific.

To formally define the quality, let $\mathcal{B}_H$ denote the subgraph of the worker-task assignment graph restricted to honest workers H and $\boldsymbol{y} = (y_1, y_2, \ldots, y_m)$ denote the vector of true labels for the tasks. Because the number of affected tasks depends on the actual honest worker responses, we focus on the case when all the honest workers are perfectly reliable, that is, they always provide the correct response. Focusing on completely reliable honest workers allows us to isolate the impact of adversaries because any misidentification is caused by the presence of adversaries. Finally, let $\mathcal{S}_k$ denote the strategy space of $k < |H|$ adversaries, where each strategy $\sigma \in \mathcal{S}_k$ specifies the $k \times m$ response matrix given by the adversaries. Because we do not restrict the adversary strategy in any way, it follows that $\mathcal{S}_k = \{-1, 0, +1\}^{k \times m}$. The *quality* of a decision rule $\boldsymbol{R} \in \mathcal{C}$ is then defined as

$$\mathrm{Aff}(\boldsymbol{R}, \mathcal{B}_H, k) = \max_{\sigma \in \mathcal{S}_k, \boldsymbol{y} \in \{-1,+1\}^m} \left| \{t_j \in \mathcal{T} \colon R_{t_j}^{\boldsymbol{y},\sigma} \neq y_j)\} \right|$$

where $R_t^{\boldsymbol{y},\sigma} \in \{-1, +1\}$ is the label output by the decision rule $\boldsymbol{R}$ for task t when the true label vector is $\boldsymbol{y}$ and the adversary strategy is σ. Note that our notation $\mathrm{Aff}(\boldsymbol{R}, \mathcal{B}_H, k)$ makes the dependence of the quality measure on the honest worker subgraph $\mathcal{B}_H$ and the number of adversaries k explicit.

We obtain the following result, which establishes a lower bound on the quality of any decision rule:

Theorem 2.10 (Lower bound on number of affected tasks)**.** *Suppose that $|A| = k$ and all honest workers provide correct responses. Let* $\mathrm{PreIm}(\mathcal{T}')$ *denote the set of honest workers who label at least one task in $\mathcal{T}' \subseteq \mathcal{T}$. Then, given any honest worker-task assignment graph $\mathcal{B}_H$, there exists an adversary strategy $\sigma^* \in \mathcal{S}_k$ that is independent of any decision rule $\boldsymbol{R} \in \mathcal{C}$, such that*

$$\max_{\boldsymbol{y} \in \{-1,+1\}^m} \mathrm{Aff}(\boldsymbol{R}, \sigma^*, \boldsymbol{y}) \geq L \quad \forall \boldsymbol{R} \in \mathcal{C}, \quad \textit{where}$$

$$L := \frac{1}{2} \cdot \left(\max_{\substack{\mathcal{T}' \subseteq \mathcal{T}: \\ |\mathrm{PreIm}(\mathcal{T}')| \leq k}} |\mathcal{T}'| \right)$$

and $\mathrm{Aff}(\boldsymbol{R}, \sigma^*, \boldsymbol{y})$ *denotes the number of affected tasks under adversary strategy* σ^**, decision rule* $\boldsymbol{R}$*, and true label vector* $\boldsymbol{y}$ *(with the assumption that the maximum over an empty set is zero). In particular, this means that* $\mathrm{Aff}(\boldsymbol{R}, \mathcal{B}_H, k) \geq L$ *for all decision rules* $\boldsymbol{R} \in \mathcal{C}$*. In addition, there exist honest worker-task assignment graphs such that* σ^* *renders the adversaries* A *indistinguishable from a set of* k *honest workers, so that a random guess misclassifies* $2k$ *workers with probability* $(1/2)$*.*

We describe the main idea of the proof. The proof proceeds in two steps: (i) we provide an explicit construction of adversary strategy σ^* that depends only on $\mathcal{B}_H$ and (ii) we show the existence of two possible true labelings $\tilde{\boldsymbol{y}} \neq \boldsymbol{y}$ such that $\boldsymbol{R}$ outputs exactly the same labels in both scenarios. The adversary labeling strategy we construct uses the idea of *indistinguishability*, which captures the fact that by carefully choosing their responses, adversaries can render themselves indistinguishable from honest workers. In the simple case when there is only one honest worker, the adversary simply flips the response provided by the honest worker, so that each task will have two labels of opposite parity. It can be argued that because there is no other discriminatory information, it is impossible for any decision rule $\boldsymbol{R} \in \mathcal{C}$ to distinguish the honest worker from the adversary and to therefore identify the true label of any task (better than a random guess). We extend this to the general case, where the adversary "targets" at most k honest workers and derives a strategy based on the subgraph of $\mathcal{B}_H$ restricted to the targeted workers. The resultant strategy can be shown to result in incorrectly identified labels for at least L tasks for some ground-truth label assignment.

Note that Theorem 2.10 holds for *any* honest worker-task assignment graph $\mathcal{B}_H$. This is particularly remarkable given that the analysis of aggregation algorithms becomes extremely complicated for general graphs (a fact observed in previous studies; see [DDKR13]). The bound L itself depends on the structure of $\mathcal{B}_H$, and therefore, it can be difficult to interpret in general. However, it becomes interpretable for (r, γ, α)-bipartite expanders, as defined next.

Definition 2.4 (Expanders)**.** An honest worker-task assignment graph $\mathcal{B}_H = (H \cup \mathcal{T}; E)$, with edges E between the honest workers H and tasks $\mathcal{T}$, is (r, γ, α)-bipartite expander if (i) $\mathcal{B}_H$ is r-right regular, that is, each task is labeled by r honest workers and (ii) for all $\mathcal{T}' \subseteq \mathcal{T}$ such that $|\mathcal{T}'| \leq \gamma |\mathcal{T}|$, the pre-image of $\mathcal{T}'$ satisfies $|\mathrm{PreIm}(\mathcal{T}')| \geq \alpha |\mathcal{T}'|$, where $\mathrm{PreIm}(\mathcal{T}')$ is the set of all honest workers who label at least one task in $\mathcal{T}'$.

Note that the definition entails that $\alpha \leq r$. We have the following corollary of Theorem 2.10 when $\mathcal{B}_H$ is (r, γ, α)-bipartite expander.

Corollary 2.10.1 (Lower bound for expanders)**.** *Suppose* $\mathcal{B}_H$ *is* (r, γ, α)*-bipartite expander. Then,* k *adversary identities can affect at least* L *tasks such that* $\lfloor \frac{k}{r} \rfloor \leq 2L \leq \lceil \frac{k}{\alpha} \rceil$*, provided* $\lceil \frac{k}{\alpha} \rceil + 1 < \gamma \cdot |\mathcal{T}|$*. Furthermore, given any constant* r*, there exists* $\gamma > 0$ *such that a uniformly random* $\mathcal{B}_H$ *is* $(r, \gamma, r-2)$*-bipartite expander with probability at least* $1/2$*, in which case the lower bound* $L = \frac{1}{2} \lceil \frac{k}{r-2} \rceil$*.*

The proof is provided in Appendix B.4. The above statement says that if the honest worker-task assignment graph $\mathcal{B}_H$ is constructed randomly, then k adversary identities can

affect at least $\frac{1}{2}\lfloor\frac{k}{r}\rfloor$ tasks. The bound implies that the ability of the adversaries to affect tasks increases linearly as the number of identities k increases. Further, the damage that k adversaries can do decreases inversely with the number of honest workers r who provide labels for each task. Both implications are intuitive. As can be seen from the proof, the lower bound $\frac{1}{2}\lfloor\frac{k}{r}\rfloor$ on L in Corollary 2.10.1 holds for all r-right regular graphs, even if they are not expanders.

Having established the above lower bound and in light of Lemma 2.9, the natural question to ask is as follows: does there exist a label aggregation algorithm for which we can prove an upper bound on the number of affected tasks, irrespective of the strategy employed by the sophisticated adversaries? Below, we show such a label aggregation algorithm that is a natural extension of the hard-penalty algorithm.

Accuracy of the hard-penalty algorithm. We introduce the *penalty-based label aggregation* algorithm (see Algorithm 7) for our analysis, which is a natural extension of the hard-penalty algorithm to also perform label aggregation:

Algorithm 7 PENALTY-BASED LABEL AGGREGATION

1: **Input:** W, $\mathcal{T}$, and $\mathcal{L}$

2: Perform steps 2 and 3 of the hard-penalty algorithm

3: For each task t_j, let $w_{t_j^+}, w_{t_j^-}$ be worker nodes that task nodes t_j^+, t_j^- are respectively mapped to in the optimal semi-matching $\mathcal{M}$

4: **Output**

$$\hat{y}_j = \begin{cases} +1 & \text{if } deg_{\mathcal{M}}(w_{t_j^+}) < deg_{\mathcal{M}}(w_{t_j^-}) \\ -1 & \text{if } deg_{\mathcal{M}}(w_{t_j^+}) > deg_{\mathcal{M}}(w_{t_j^-}) \\ \leftarrow \{-1, +1\} & \text{otherwise} \end{cases}$$

(here $\hat{y}_j$ refers to the output label for task t_j and $\leftarrow \{-1, +1\}$ means that $\hat{y}_j$ is drawn uniformly at random from $\{-1, +1\}$)

For our analysis, we consider the following model for honest worker responses that is based on the *spammer-hammer* model popularly used in previous studies [KOS11, LPI12, OOSY16, KO16]: $0 \leq \varepsilon < 1$ fraction of honest workers are "spammers," that is, they make mistakes in their responses, and the remaining $1 - \varepsilon$ fraction are "hammers," that is, they are perfectly reliable and always provide the correct response. In Theorem 2.11 below, $\mathcal{B}_H^{cs}$ refers to the bipartite graph created as follows: (i) each honest worker h is represented by a node on the left; (ii) each task $t_j \in \mathcal{T}$ is represented by (at most) two nodes on the right, t_j^+ and t_j^-; and (iii) add the edge (h, t_j^+) (resp. (h, t_j^-)) if honest worker h labels task t_j as $+1$ (resp. -1). Then, we can show the following:

Theorem 2.11 (Accuracy of penalty-based label aggregation)**.** *Suppose that* $|A| = k$ *and let* $\varepsilon \in [0, 1)$ *denote the fraction of honest workers who make mistakes in their responses. Furthermore, let* $d_1 \geq d_2 \geq \cdots \geq d_{|H|}$ *denote the degrees of honest workers in the optimal semi-matching on* $\mathcal{B}_H^{cs}$*. For any true labeling* $\boldsymbol{y}$ *of the tasks and under Algorithm 7 (with the convention that* $d_i = 0$ *for* $i > |H|$*):*

1. *If* $\varepsilon = 0$*, there exists an adversary strategy* σ^* *such that the number of affected tasks is at least* $\frac{1}{2}\sum_{i=1}^{k-1} d_i$
2. *Assuming that each task receives at least one correct response from the honest workers, no adversary strategy can affect more than* U *tasks where*
   (a) $U = \sum_{i=1}^{k+\varepsilon\cdot|H|} d_i$*, when at most one adversary provides correct responses*
   (b) $U = \sum_{i=1}^{2k+\varepsilon\cdot|H|} d_i$*, in the general case*

A few remarks are in order. First, it can be shown that for any two optimal semi-matchings for the bipartite graph $\mathcal{B}_H^{cs}$, the degree sequence $d_1, d_2, \ldots, d_{|H|}$ is the same and therefore, the bounds in the result above are uniquely defined given $\mathcal{B}_H^{cs}$. The result of Theorem 2.11 provides both a lower and upper bound for the number of tasks that can be affected by k adversaries under the penalty-based label aggregation algorithm, irrespective of the adversary strategy. This is remarkable, especially because we established that existing label aggregation algorithms can be arbitrarily bad (Lemma 2.9). Assuming honest workers are always correct, that is, $\varepsilon = 0$, our characterization is reasonably tight when all but (at most) one adversary provide incorrect responses. In this case, the gap between the upper and a constant factor of the lower bound is d_k, which can be "small" for large enough k. However, our characterization is loose in the general case when adversaries can provide arbitrary responses. Here, the gap is $\sum_{i=k}^{2k} d_i$; we attribute this to our proof technique and conjecture that the upper bound of $\sum_{i=1}^{k} d_i$ also applies to the more general case. When $\varepsilon > 0$, the upper bound U increases because there are more incorrect responses and, in turn, the scope for adversaries to affect a larger number of tasks.

One might wonder whether we could perform a similar analysis for the case when honest workers follow the one-coin model. Given the complexity of analyzing optimal semi-matchings, our current proof technique does not readily extend to this scenario, and therefore, there is an opportunity to improve the analysis in future work.

Optimality of penalty-based label aggregation. We now compare the upper bound U in Theorem 2.11 to the lower bound L in Theorem 2.10 in the case when honest workers are perfectly reliable, that is, $\varepsilon = 0$. We show that when the degrees $d_1, d_2, \ldots, d_{|H|}$ are all distinct, $L \geq \frac{1}{2}\sum_{i=1}^{k-1} d_i$. Combined with Theorem 2.10, this shows that k adversaries can affect at least $\frac{1}{2}\sum_{i=1}^{k-1} d_i$ tasks *irrespective* of the label aggregation algorithm used to aggregate the worker responses. From Theorem 2.11, we also have that under the penalty-based label aggregation algorithm, k adversaries can affect at most $U = \sum_{i=1}^{2k} d_i \leq 3(\sum_{i=1}^{k-1} d_i)$ tasks (as

long as $k \geq 2$). Therefore, our algorithm achieves constant factor optimality in recovering the true labels of tasks *irrespective* of the adversary's strategy.

2.5 Numerical Analysis

We conducted two empirical studies to demonstrate the practical value of our methods. The first study illustrates a concrete real-world application of our methodology. By using standard crowdsourcing datasets (as described below), it shows that filtering out the adversaries identified by our methods allows existing label aggregation algorithms to infer true task labels more accurately. Such improvements in accuracy by *discarding* labels from certain workers suggests that their label patterns do not conform to standard probabilistic assumptions. Although not illustrated in our study, instead of being filtered out, the adversary labels may also be used by fitting a model different from that of honest workers in applications where such assumptions are reasonable. The second study is designed to assess the ability of our methods to successfully identify adversaries and low-reliability honest workers. It is a simulation study in which we injected a standard crowdsourcing task with "spammers" (workers who label a task $+1$ and -1 with probability $1/2$ each irrespective of the true label) and workers adopting the Uniform strategy. It demonstrates that both soft- and hard-penalty algorithms successfully identify adversaries and low-reliability honest workers when the worker-task assignment graph has a power-law degree distribution for workers and tasks, thus complementing the results in Section 2.4.1 which focused on (l, r)-regular worker-task assignment graphs.

For the purposes of our studies, we focused on the following six label aggregation algorithms: (a) simple majority algorithm MV; (b) EM algorithm for the two-coin model [RY12]; (c) KOS algorithm [KOS11]; (d) KOS(NORM), a normalized variant of KOS in which messages are scaled by the corresponding worker and task degrees to account for non-regular node degrees in the assignment graph; (e) SPEC-EM, the spectral EM algorithm of [ZCZJ14]; and (f) MMCE, the regularized minimax conditional entropy approach of [ZLP+15]. We implemented both variants of our reputation algorithm: (a) soft-penalty (SOFT) and (b) hard-penalty (HARD). As removing workers alters the penalties of the remaining workers, we filtered the workers iteratively. In each iteration, we recomputed the penalties of the remaining workers, removed the worker with the highest penalty, and repeated until a prespecified number of workers was removed.[5]

Finally, as mentioned in Section 2.4.1, we implemented two variants of the soft-penalty algorithm: one in which non-conflict tasks are dropped for assigning worker penalties and another in which they are retained. The results were essentially the same, so we only report the results for the variant in which the non-conflict tasks were dropped.

[5]The performance was similar for the non-iterative variant discussed in Section 2.3—we report the results for the iterative version because it had marginally better performance.

Dataset	Workers	Tasks	Responses
rte	164	800	8000
temp	76	462	4620
stage2	68	711	2035
task2	386	2269	12435
tweets	66	1000	4977

Table 2.1: Statistics of real datasets used in the experiments.

2.5.1 Accuracy improvements on real crowdsourced datasets

We considered the following standard datasets:

- `stage2` and `task2` [TL11]: consisting of a collection of topic-document pairs labeled as relevant or non-relevant by workers on Amazon Mechanical Turk. These datasets were collected as part of the TREC 2011 crowdsourcing track.
- `rte` and `temp` [SOJN08]: consisting of annotations by Amazon Mechanical Turk workers for different natural language processing (NLP) tasks. `rte` consists of binary judgments for textual entailment (whether one sentence can be inferred from another) and `temp` for temporal ordering of events.
- `tweets` [MSF+14]: consisting of sentiment (positive or negative) labels for 1K tweets.

As inferring the reliabilities of workers who labeled very few tasks[6] is difficult, we preprocessed the datasets to remove all workers who labeled less than three tasks. Table 2.1 shows summary statistics of the datasets after our preprocessing.

Table 2.2 reports the accuracies of various label aggregation algorithms for inferring true task labels. For each benchmark label aggregation algorithm, the column BASE reports the accuracy of the algorithm in isolation. The columns SOFT and HARD report the best accuracy of the algorithms for filtering $k = 1, 2, \ldots, 10$ workers using the soft- and hard-penalty algorithms, respectively; the numbers in parentheses indicate the k value for which we observed the best performance.[7]

The key conclusion we draw is that filtering out workers flagged by our algorithms as adversaries boosts the predictive accuracy of state-of-the-art aggregation algorithms significantly across the datasets: the average improvement in accuracy for MV is 3.7%, EM is 3.4%, KOS is 30.2%, KOS(NORM) is 4.4%, SPEC-EM is 12.6%, and MMCE is 4.7% when using the hard-penalty algorithm. The improvement is large for KOS because it is designed for

[6]The datasets `stage2`, `task2`, and `tweets` contain several workers who provided responses for only a single task.

[7]The performance was robust to the choice of k. We matched or improved the accuracy of the underlying label aggregation algorithm in 66% and 70% of cases on average for the soft- and hard-penalty algorithm, respectively.

Percentage accuracy in recovering true labels of benchmark algorithms in isolation, and when combined with our (modified) reputation algorithms

	rte			temp			stage2			task2			tweets		
Algorithm	BASE	SOFT	HARD	BASE	SOFT	HARD	BASE	SOFT	HARD	BASE	SOFT	HARD	BASE	SOFT	HARD
MV	91.9	92.1(7)	**92.5(3)**	93.9	93.9	**94.4(5)**	74.3	75.4(1)	**80.5(2)*****	64.2	64.2	**67.8(10)*****	69.6	69.8(4)	**73.3(1)*****
EM	93.0	93.0	**93.3(5)**	94.1	94.1	94.1	70.2	76.8(4)	**81.2(6)*****	67.0	67.1(6)	**68.6(9)*****	71.2	71.2	**71.7(1)**
KOS	49.7	89.0(10)	**91.6(10)*****	56.9	69.3(4)	**93.7(3)*****	74.5	74.7(7)	**75.1(2)**	57.4	57.4	**65.6(10)*****	65.8	66.0(4)	**70.5(1)*****
KOS(NORM)	91.3	92.6(5)	**93.1(6)****	93.9	94.4(7)	**94.4(1)**	75.5	75.5	**78.2(2)***	58.3	58.9(8)	**67.7(9)*****	68.7	68.7	**71.0(2)*****
SPEC-EM	90.1	92.4(8)	**92.8(6)**	56.1	94.2(7)	**94.4(5)**	82.8	82.8	82.8	56.1	56.1	**57.6(10)*****	67.0	67.0	**69.0(1)*****
MMCE	92.5	92.9(9)	**93.2(3)**	94.4	94.4	94.4	57.0	58.5(5)	**73.3(10)*****	66.1	66.1	**66.9(10)**	72.7	72.7	72.7

Table 2.2: For each benchmark, the best-performing combination is highlighted in bold. The number in parentheses represents the number of workers filtered by our reputation algorithm (an absence indicates that no performance improvement was achieved while removing upto 10 workers with the highest penalties).
The improvements are marked significant according to a paired samples t-test at * 10%, ** 5%, and *** 1% significance levels.

regular graphs (with all workers having the same degree and all tasks having the same degree) and suffers in performance on real-world graphs that are not regular. Second, we note that our methods can boost the performance of the MV and KOS algorithms to the level of the popular EM algorithm. The MV algorithm is simple to implement, and the KOS algorithm is designed for scenarios where the underlying assignment graph is random (l, r)-regular and has strong theoretical guarantees and robustness to different initializations [KOS14]. Our results suggest that implementing the MV and KOS algorithms in conjunction with our reputation algorithms can allow us to obtain their respective simplicity and robustness along with strong practical performance (even for irregular graphs) comparable to that of the EM algorithm.[8] Finally, because discarding the labels of certain workers improves the predictive accuracy, our results suggest that standard probabilistic models (including the two-coin model) are insufficient to capture the labeling patterns of workers in real-world datasets.

To gain insights into the types of workers identified by our algorithms, we conducted a qualitative analysis of the labeling patterns of the workers that were filtered out. We observed the following key types:

1. Workers who labeled at least 10 tasks, of which more than 95% had the same label. For instance, our algorithms detected six such workers in the `temp` dataset, five in the `task2` dataset, and one in the `tweets` dataset. For the `stage2` dataset, we detected two workers who gave all $+1$ labels and one worker who gave all but a single response as -1; it should be noted that the empirically calculated prevalence γ of $+1$ tasks in the `stage2` dataset was 0.83, potentially suggesting that the two workers who gave all $+1$ labels were adopting "smart" strategies.

2. Workers who provide labels independent of true task labels. For instance, we detected four such workers in the `rte` dataset, seven workers in the `temp` dataset, seven in the `task2` dataset, three in the `stage2` dataset, and one in the `tweets` dataset, whose label patterns are such that the empirical fractions $\hat{\alpha}$ and $\hat{\beta}$ of correct responses among the tasks with true labels $+1$ and -1, respectively, satisfy $|\hat{\alpha} + \hat{\beta} - 1| \leq 0.05$. [RY12] showed that such workers effectively assign a label of $+1$ with probability $\hat{\alpha}$ and -1 with probability $1 - \hat{\alpha}$ independent of the true task label.

3. Workers with skewed reliabilities. Such workers were accurate on tasks with one type of true label, say, $+1$, but not on others, say, tasks with true label -1. Such label patterns of workers may be indicative of tasks that require subjective assessments. For instance, we found four workers in the `tweets` dataset and two workers in `stage2` dataset that had skewed reliabilities. In the `tweets` dataset, workers were asked to rate the sentiment of a tweet as being positive or negative. As the notion of tweet sentiment can be subjective, workers with biased views of the sentiment make systematic errors on one type of task. A similar explanation applies to the `stage2` dataset, in which workers

[8] Note that the EM algorithm can also have theoretical guarantees with appropriate initializations (see [GZ16] and [ZCZJ14]).

were asked to label a topic-document pair as relevant or not, requiring a potentially subjective assessment. See [KKH15] for more examples.

In summary, our reputation algorithms are successful in identifying adversarial workers adopting a broad set of strategies. Furthermore, although not reported, the empirical reliabilities of the workers filtered out using our algorithms were on average *lower* than those of unfiltered workers. This suggests that the labels our algorithms discard are from low-reliability workers who provide little to no information about the true task labels; this explains the improved accuracy we obtain.

2.5.2 Identifying low-reliability honest workers and adversaries

We use a simulation study to show that our reputation algorithms successfully identify low-reliability honest workers and adversaries when the worker-task assignment graphs have power-law degree distributions for the worker and task nodes. Such graph structures are common in many real-world crowdsourcing scenarios [FKK+11]. The results of the simulation study complement the theoretical results presented in Section 2.4.1 for (l, r)-regular assignment graphs, which show that the soft-penalty algorithm successfully identifies low-reliability honest workers and adversaries adopting the Uniform strategy.

For our study, we used the following broad procedure: (a) generate a random crowdsourcing instance from the ground-truth model, (b) generate synthetic responses from the workers for a sample of tasks, (c) filter out workers with the highest penalties according to our reputation algorithms, and (d) compute the *precision*, that is, the fraction of filtered-out workers who are adversarial and who have low empirical reliabilities.

Setup. We considered a total of $n = 100$ workers. The probability q that a worker is honest was set to 0.7; therefore, on average, there are 30 adversaries among the 100 workers. The prevalence γ of $+1$ tasks was set to 0.5. We sampled worker degrees according to a power-law distribution (with exponent $a = 2.5$) with the minimum degree equal to 5, and then, we used the Python networkx library [HSS08] to generate the worker-task assignment graph.[9] Note that the number of tasks m is determined from the total number of workers and the sampled worker degrees.

As the worker degrees are skewed, the performance of the algorithms is influenced by the adversary degrees. To capture this, we considered two scenarios: (a) adversaries have high degrees and (b) adversaries have low degrees, and biased the probability of a worker being an adversary according to her degree. See Appendix B.6 for the details.

For each scenario, after the worker-task assignment graph and worker identities were sampled, we generated the crowdsourcing instances as follows: (a) set the true label y_j of each task t_j to be $+1$ with probability $1/2$ and -1 with probability $1/2$; (b) for each honest worker w, sample its reliability μ_w u.a.r from the interval $[0.8, 1.0)$, and set its response to

[9]Specifically, we used the following function: https://networkx.github.io/documentation/networkx-1.10/reference/generated/networkx.algorithms.bipartite.generators.preferential_attachment_graph.html.

Precision of reputation algorithms when filtering 10 workers

	Spammer		Uniform	
	Low Deg	High Deg	Low Deg	High Deg
SOFT	**90.03**	88.70	**77.13**	69.57
HARD	89.00	**93.83**	73.43	**76.73**

Table 2.3: The rows SOFT and HARD correspond to the soft- and hard-penalty algorithms, respectively. Spammer and Uniform correspond to the two adversary strategies, and "Low Deg" and "High Deg" refer to the scenarios in which low- and high-degree workers, respectively, are more likely to be adversaries. Refer to the text for more details.

each assigned task t_j to be the true label y_j with probability μ_w and $-y_j$ with probability $1-\mu_w$; and (c) generate responses from adversarial workers according to the chosen strategy. We focused on two adversary strategies: (a) Spammer—label each task $+1$ or -1 with prob. $1/2$ and (b) Uniform—label every assigned task $+1$. The first strategy reflects the setting of Theorem 2.2 because it is captured by the one-coin model, and the second strategy reflects the setting of Theorem 2.3.

Results and Discussion. Table 2.3 reports the *precisions* of the SOFT and HARD algorithms when (iteratively) removing 10 workers. The precision is defined as the fraction of filtered-out workers who are either adversarial or honest with empirical reliabilities less than 0.85 (which is less than the mean $\mu = 0.9$ of the honest worker reliability distribution), where the empirical reliability of an honest worker is equal to the fraction of its responses that were correct. The table reports the precision of the algorithms for the two adversary strategies, Spammer and Uniform, under the two scenarios: *Low Deg*, in which low-degree workers are more likely to be adversaries, and *High Deg*, in which high-degree workers are more likely to be adversaries. For each combination of adversary strategy and scenario, the reported numbers are the precision values averaged over 300 randomly generated instances.

We draw the following key conclusions. First, our algorithms have precision values $> 69\%$ in all scenarios, indicating that they are successful in identifying adversaries and low-reliability honest workers when the worker and task degrees have a power-law distribution. This finding complements the results of Theorem 2.4 and Theorem 2.7, which establish a similar result when the worker-task assignment graph is (l, r)-regular. Second, our results offer insights into the settings under which the soft- or hard-penalty algorithm is appropriate. Generally, we observe that the hard-penalty algorithm is more appropriate when the adversaries have higher degrees, whereas the soft-penalty algorithm is more appropriate when the adversaries have lower degrees. We also note that when the adversaries have labeling patterns (such as Spammer) that are probabilistically similar to those of honest workers, the soft-penalty algorithm has a performance comparable to that of the hard-penalty algorithm even when the adversaries have high degrees.

Part II

Demand Learning

Chapter 3

Nonparametric estimation of mixture of logit models

3.1 Introduction

Mixture models are used for modeling a wide-range of phenomena in many fields. Within operations, they have been used to model customer demand, which changes in response to the changes a firm makes to its product offerings. Predicting these changes allows firms to optimize their product and price offerings, such as discontinuing low demand products or enforcing price changes to shift demand to specific products. Demand predictions also serve as key inputs to inventory control and price optimization models that are used in retail operations and revenue management (RM) systems. A typical prediction problem involves fitting a mixture model to historical sales transactions and inventory data. The most popular model that is fit is the mixture of multinomial logit (MNL) models, also known simply as the mixture of logit models. This model has received considerable attention in the literature and has also been successfully applied in practice. In addition, it has been shown to approximate a wide class of mixtures [MT00].

Because of its significance for demand modeling, we focus on the problem of estimating the mixing distribution of a mixture of logit models, from sales transaction and inventory data. The main challenge in this problem is that the structure of the mixing distribution is not known in practice. A common work-around is to assume that the mixing distribution comes from a pre-specified *parametric* family, such as the normal or the log-normal distribution, and then estimate the parameters via maximum likelihood estimation [Tra09]. This approach is reasonable when there is some prior knowledge about the structure of the underlying mixing distribution. But when no such knowledge exists, as often happens in practice, the ground-truth mixing distribution may very well not conform to the imposed parametric form. This leads to model misspecification, which can result in biased parameter estimates [Tra08] or low goodness-of-fit measures [FKRB11].

To avoid model misspecification, we take a nonparametric approach, in which we search for the best fitting mixing distribution from the class of all possible mixing distributions.

The challenge with this approach is a computational one. The class of all possible mixing distributions lacks sufficient structure to allow for tractable estimation methods. One approach in the literature has been to approximate the class of all possible mixing distributions with another (large) class, and then search for the best fitting mixing distribution in that approximate space. For instance, [Tra08] takes this approach, in which the space of all mixing distributions is approximated with the class of finite mixtures of normal distributions or the class of discrete distributions with a large support size. Such approximations allow the application of standard optimization techniques, such as the expectation-maximization (EM) framework. But the resulting optimization problems are still non-convex and become difficult to solve, running into numerical issues, as the number of parameters increases.

Our main contribution in this chapter is to reformulate the nonparametric mixture estimation problem as a constrained convex program, *without* resorting to any approximations to the space of all possible mixing distributions. We pose the mixture estimation problem as the problem of searching for the distribution that minimizes a loss function from among the class of all possible mixing distributions. The standard log-likelihood loss (which results in the maximum likelihood estimator) and squared loss are two example loss functions. Then, we use the insight that the mixing distribution affects the objective function only through the choice probabilities it predicts for the observed choices in the data; we call the vector of these choice probabilities, the *data mixture likelihood vector*. Now, instead of optimizing over the space of mixing distributions, the mixture estimation problem can be solved by directly optimizing over the space of all possible data mixture likelihood vectors. The constraints ensure that the mixture likelihood vector is indeed consistent with a valid mixing distribution. We show that for the standard loss functions used in the literature, the objective function is convex in the mixture likelihood vector. Further, although not apriori clear, we show that the constraint set is also convex. Together, these two properties result in a constrained convex program formulation for the mixture estimation problem. We emphasize here that although we obtain a convex program, the constraint space lacks an efficient description. Therefore, the resulting program, though convex, may be theoretically hard to solve. Nevertheless, there is vast literature on solving such convex programs, which we leverage to obtain scalable and numerically stable methods that are efficient for special cases and result in good approximations more generally.

A more immediate concern is that simply solving the above program is not sufficient because the optimal solution will be expressed as the mixture likelihood vector and not as the mixing distribution. Backing out the underlying mixing distribution from the mixture likelihood vector may again be a computationally intensive exercise. To counter this issue, we apply the conditional gradient (a.k.a. Frank-Wolfe) algorithm to solve the above convex program. We show that the special structure of the conditional gradient (CG) algorithm allows it to simultaneously perform both the tasks of optimizing over the predicted choice probabilities *and* recovering the best fitting mixing distribution. The CG algorithm is an iterative first-order method for constrained convex optimization. We show that when applied to our method, each iteration of the CG algorithm yields a single mixture component. The CG algorithm has seen an impressive revival in the machine learning literature recently because

of its favorable properties compared to standard projected/proximal gradient methods, such as efficient handling of complex constraint sets. The vast literature on the CG algorithm in the machine learning area confers two key advantages to our estimation technique: (a) availability of precise convergence guarantees [LJJ15] and (b) scalability to large-scale and high-dimensional settings [WSD+16].

3.1.1 Relevant literature

Our work has connections to two broad areas:

Nonparametric maximum likelihood estimation (NPMLE). Our estimation approach generalizes the NPMLE techniques, which have a long and rich history in classical statistics [Rob50, KW56]. These techniques search for a distribution that maximizes the likelihood function from a large class of mixing distributions. In the context of studying properties of the maximum likelihood estimator (such as existence, uniqueness, support size, etc.) for the mixing distribution via the geometric structure of the constraint set, [Lin83] shows that when the mixing distribution is unrestricted, the NPMLE can be formulated as a convex program. However, such a formulation is computationally difficult to solve when the underlying parameter space is high dimensional. To address this issue, existing work has taken two approaches. The first approach reduces the search space to a large (but finite) number of mixture components, and uses the EM algorithm for estimation [Lai78]. Though now the estimation problem is finite-dimensional, convexity is lost and standard issues related to non-convexity and finite mixture models become a significant obstacle [MP00]. The second approach retains convexity but gains tractability through a finite-dimensional convex approximation where the support of the mixing distribution is assumed to be finite and pre-specified (such as a uniform grid) and only the mixing weights need to be estimated. Fox et al. [FKRB11] specialize this approach to estimating a mixture of logit models. However, it is unclear how to choose the support. When the dimensionality of the parameter space is small, the authors demonstrate that a uniform grid is sufficient to reasonably capture the underlying distribution but this approach quickly becomes intractable for even moderately large parameter dimensions.[1] Consequently, existing techniques have usually focused on simple models with univariate or low-dimensional (bi- and tri-variate) mixing distributions [BSL92, JZ09, FD18] to retain tractability.

In the context of the above, we avoid the issues resulting from the non-convex formulation by retaining convexity, but at the same time we do *not* need access to a pre-specified support. We leverage the conditional gradient algorithm to directly solve the seemingly intractable convex program, which iteratively generates the support of the mixing distribution by searching over the underlying parameter space. This allows our method to scale to higher-dimensional settings, 5 in our SUSHI case study and 11 in the IRI case study; see Sections 3.6.1 and 3.6.2.

Conditional gradient algorithms. The conditional gradient algorithm is one of the earliest methods [FW56] for constrained convex optimization, and has recently seen an impressive revival for solving large-scale problems with structured constraint sets (see [Cla10] and [Jag11]

[1] Indeed, their numerical experiments focus on only bivariate mixing distributions.

for excellent overviews). The algorithm has been used in diverse domains including computer vision [JTFF14], submodular function optimization [Bac13], collaborative filtering [JS10], as well as inference in graphical models [KLJS15]. In addition, numerous related variants of the algorithm have been proposed such as solving non-linear subproblems to increase sparsity [Zha03] and incorporating regularization to improve predictive performance [HJN15]. In terms of theoretical performance, Jaggi [Jag13] gave a convergence analysis that guarantees an error of at most $O(1/t)$ (sublinear convergence) after t iterations for any compact convex constraint set. Recently, Lacoste-Julien and Jaggi [LJJ15] proved that many versions of the classical Frank-Wolfe algorithm enjoy global linear convergence for any strongly convex function optimized over a polytope domain.

Our main contribution is leveraging the conditional gradient algorithm for estimating the mixing distribution. We also provide a novel theoretical analysis for convergence in the case of log-likelihood loss (see Section 3.4.1), where existing results are not applicable since the gradient is unbounded at the boundary of the constraint domain. We also show that, under appropriate structure in the product feature space, the algorithm converges to the optimal solution in a *finite* number of iterations (see Section 3.4.3).

3.2 Problem Setup and Formulation

We consider a universe $[n] \stackrel{\text{def}}{=} \{1, 2, \ldots, n\}$ of n products,[2] which customers interact with over $T \geq 1$ discrete time periods. In each time period $t \in [T]$, the firm offers a subset $S_t \subseteq [n]$ of products to the customers and collects sales counts for each of the products. We let N_{jt} denote the number of times product j was purchased in period t, $N_t \stackrel{\text{def}}{=} \sum_{j \in S_t} N_{jt}$, the total number sales in period t, and $N \stackrel{\text{def}}{=} \sum_{t \in [T]} N_t$, the total number of sales over all the time periods. We suppose that we observe at least one sale in each period t, so that $N_t > 0$ for all periods $t \in [T]$; if there was no observed sale in a time period, then we assume that it was already dropped from the observation periods. Let $\mathsf{Data} \stackrel{\text{def}}{=} \{(N_{jt} \colon j \in S_t) \mid t \in [T]\}$ denote all the observations collected over the T discrete time periods. We assume that product $j \in S_t$ is represented by a D-dimensional feature vector $\mathbf{z}_{jt}$ in some feature space $\mathcal{Z} \subseteq \mathbb{R}^D$. Example features include price, brand, and color. Product features could vary over time; for instance, product prices may vary because of promotions, discounts, etc. In practice, these data are often available to firms in the form of purchase transactions, which provide sales information, and inventory data, which provide offer-set information.

We assume that each customer makes choices according to an MNL (aka logit) model, which specifies that a customer purchases product j from offer-set S with probability

$$f_{j,S}(\boldsymbol{\omega}) = \frac{\exp\left(\boldsymbol{\omega}^\top \mathbf{z}_{jS}\right)}{\sum_{\ell \in S} \exp\left(\boldsymbol{\omega}^\top \mathbf{z}_{\ell S}\right)}, \tag{3.1}$$

[2] We use the notation $[m] \stackrel{\text{def}}{=} \{1, 2, \ldots, m\}$ for any positive integer m in the rest of the chapter.

where $\mathbf{z}_{\ell S}$ is the feature vector of product ℓ when offered as part of offer-set S and $\boldsymbol{\omega}$ is the parameter or "taste" vector. This taste vector specifies the "value" that a customer places on each of the product features in deciding which product to purchase. Customers often have heterogeneous preferences over product features. To capture this heterogeneity, we assume that the population of customers is described by a mixture of MNL models, where in each choice instance, a customer samples a vector $\boldsymbol{\omega}$ according to some distribution Q (over the parameter space $\mathbb{R}^D$) and then makes choices according to the MNL model with parameter vector $\boldsymbol{\omega}$.

Our goal is to estimate the best fitting mixture distribution Q to the collection Data of sales observations. Before we state our mixture estimation problem formally, we describe the traditional approaches to mixture estimation, and their limitations, to motivate the need for our approach.

3.2.1 Traditional approaches to mixture estimation

The traditional approach assumes that the mixing distribution belongs to a family $\mathcal{Q}(\Theta)$ of distributions parametrized via parameter space Θ, where $\mathcal{Q}(\Theta) \stackrel{\text{def}}{=} \{Q_{\boldsymbol{\theta}} \colon \boldsymbol{\theta} \in \Theta\}$ and $Q_{\boldsymbol{\theta}}$ is the mixing distribution corresponding to the parameter vector $\boldsymbol{\theta} \in \Theta$. The best fitting distribution is then obtained by solving the following likelihood problem:

$$\min_{\boldsymbol{\theta} \in \Theta} \; -\sum_{t=1}^{T} \sum_{j \in S_t} N_{jt} \log \left(\int f_{jt}(\boldsymbol{\omega}) \; dQ_{\boldsymbol{\theta}}(\boldsymbol{\omega}) \right), \tag{3.2}$$

where for brevity of notation, we let $f_{jt}(\boldsymbol{\omega})$ denote $f_{j,S_t}(\boldsymbol{\omega})$. Different assumptions for the family $\mathcal{Q}(\Theta)$ result in different estimation techniques.

The most common assumption is that the mixing distribution follows a multivariate normal distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, parametrized by $\boldsymbol{\theta} = (\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu}$ is the mean and $\boldsymbol{\Sigma}$ is the variance-covariance matrix of the distribution. The resulting model is referred to as the random parameters logit (RPL) model [Tra09], and the corresponding likelihood problem is given by

$$\min_{\boldsymbol{\mu}, \boldsymbol{\Sigma}} -\sum_{t=1}^{T} \sum_{j \in S_t} N_{jt} \log \left(\int f_{jt}(\boldsymbol{\omega}) \cdot \frac{1}{\sqrt{(2\pi)^D \, |\boldsymbol{\Sigma}|}} \exp \left(-\frac{1}{2} (\boldsymbol{\omega} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\boldsymbol{\omega} - \boldsymbol{\mu}) \right) d\boldsymbol{\omega} \right).$$

The integral in the above problem is often approximated through a Monte Carlo simulation.

The other common assumption is that the mixing distribution has a finite support of size K. The distribution is then parametrized by $\boldsymbol{\theta} = (\alpha_1, \ldots, \alpha_K, \boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_K)$, where $(\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_K)$ denotes the support of the distribution and $(\alpha_1, \ldots, \alpha_K)$ denote the corresponding mixture proportions with $\sum_{k \in [K]} \alpha_k = 1$ and $\alpha_k \geq 0$ for all $k \in [K]$. The resulting model is referred to as the latent class MNL (LC-MNL) model [Bha97], and the corresponding likelihood problem is given by

$$\min_{\substack{\alpha_1, \alpha_2, \ldots, \alpha_K \\ \boldsymbol{\omega}_1, \boldsymbol{\omega}_2, \ldots, \boldsymbol{\omega}_K}} -\sum_{t=1}^{T} \sum_{j \in S_t} N_{jt} \log \left(\sum_{k=1}^{K} \alpha_k f_{jt}(\boldsymbol{\omega}_k) \right) \text{ subject to } \sum_{k \in [K]} \alpha_k = 1, \alpha_k \geq 0 \; \forall \; k \in [K].$$

Although commonly used, these traditional approaches suffer from two key limitations:

- *Model misspecification*: The most significant issue with traditional approaches is model misspecification, which occurs when the ground-truth mixing distribution is not contained in the search space $\mathcal{Q}(\Theta)$. In practice, such misspecification is common because the selection of the search space $\mathcal{Q}(\Theta)$ is often driven by tractability considerations as opposed to knowledge of the structure of the ground-truth mixing distribution. Model misspecification can result in biased parameter estimates [Tra08] and low goodness-of-fit measures [FKRB11].

- *Computational issues*: Another practical issue is that, even if the model is not misspecified, the resulting likelihood problems are non-convex and therefore hard to solve in general.

3.2.2 Our approach: mixture estimation by solving a convex program

Our approach is designed to address the challenges described above. Broadly, it has two key steps:

1. To overcome the model misspecification issue, we search over all possible mixing distributions instead of restricting our search to specific parametric families.

2. To address the computational issue, we transform the mixture estimation problem into a convex program, allowing us to tap into the vast existing literature that has proposed efficient algorithms for convex optimization.

We now describe these steps in detail and how they result in our final formulation.

Dealing with model misspecification. We arrive at our formulation through a sequence of transformations starting from the traditional mixture estimation problem in (3.2). This problem can be equivalently written as follows:

$$\min_{\boldsymbol{\theta}\in\Theta} \; -\sum_{t=1}^{T}\sum_{j\in S_t} N_{jt}\log\left(\int f_{jt}(\boldsymbol{\omega})\; dQ_{\boldsymbol{\theta}}(\boldsymbol{\omega})\right) \quad \equiv \min_{Q\in\mathcal{Q}(\Theta)} \; -\sum_{t=1}^{T}\sum_{j\in S_t} N_{jt}\log\left(\int f_{jt}(\boldsymbol{\omega})\; dQ(\boldsymbol{\omega})\right),$$

where recall that $\mathcal{Q}(\Theta) = \{Q_{\boldsymbol{\theta}} \colon \boldsymbol{\theta}\in\Theta\}$. This equivalence follows immediately from noting that the parameter vector $\boldsymbol{\theta}$ affects the objective function only through $Q_{\boldsymbol{\theta}}$. In other words, we directly search over the space of mixing distributions. To deal with model misspecification, we relax the search to be over the family of *all* possible mixing distributions and focus on the following optimization problem:

$$\min_{Q\in\mathcal{Q}} \; -\sum_{t=1}^{T}\sum_{j\in S_t} N_{jt}\log\left(\int f_{jt}(\boldsymbol{\omega})\; dQ(\boldsymbol{\omega})\right) \text{ where } \mathcal{Q} \stackrel{\text{def}}{=} \{Q \colon Q \text{ is a distribution over } \mathbb{R}^D\}.$$

The above optimization problem can be stated more compactly as

$$\min_{Q \in \mathcal{Q}} - \sum_{t \in [T], j \in S_t} N_{jt} \log \mathrm{g}_{jt}(Q),$$

where for any $t \in [T]$ and $j \in S_t$, we define the mapping $\mathrm{g}_{jt} \colon \mathcal{Q} \to [0,1]$ as

$$\mathrm{g}_{jt}(Q) = \int f_{jt}(\boldsymbol{\omega}) \, \mathrm{d}Q(\boldsymbol{\omega}).$$

That is, $\mathrm{g}_{jt}(Q)$ is the probability of choosing product j from offer-set S_t under the mixing distribution Q.

So far we have assumed that our goal is to solve the likelihood problem, or minimize, what we call, the *negative log-likelihood loss*. In practice, other loss functions are also commonly used. In order to accommodate these, we consider the following general formulation. Let $M \stackrel{\text{def}}{=} |S_1| + \cdots + |S_T|$ and $\mathbf{g} \colon \mathcal{Q} \to [0,1]^M$ denote the vector-valued mapping, defined as $\mathbf{g}(Q) = (\mathrm{g}_{jt}(Q) \colon t \in [T], j \in S_t)$. We call $\mathbf{g}(Q)$ the *data mixture likelihood vector* or simply the *mixture likelihood vector*, under mixing distribution Q. Our goal then is to solve

$$\min_{Q \in \mathcal{Q}} \ \mathsf{loss}(\mathbf{g}(Q); \mathsf{Data}), \qquad \text{(Mixture Estimation)}$$

where $\mathsf{loss}(\cdot;\ \mathsf{Data}) \colon [0,1]^M \to \mathbb{R}_+$ is a non-negative convex function. Two example loss functions include:

- **Negative log-likelihood (NLL) Loss**: This loss function was introduced earlier and is by far the most widely used in practice [Tra08]:

$$\mathsf{NLL}(\mathbf{g}(Q); \mathsf{Data}) = -\frac{1}{N} \sum_{t=1}^{T} \sum_{j \in S_t} N_{jt} \log \left(\mathrm{g}_{jt}(Q) \right).$$

- **Squared (SQ) Loss**: This loss function was employed by [FKRB11]:

$$\mathsf{SQ}(\mathbf{g}(Q); \mathsf{Data}) = \frac{1}{2 \cdot N} \sum_{t=1}^{T} N_t \cdot \sum_{j \in S_t} \left(\mathrm{g}_{jt}(Q) - y_{jt} \right)^2.$$

  where $y_{jt} \stackrel{\text{def}}{=} N_{jt}/N_t$ denotes the fraction of sales for product j in offer-set S_t.

We note that searching over the space of all possible mixing distributions can lead to potential overfit issues, which we discuss in Section 3.3.1.

Formulating mixture estimation as a convex program. We now focus on solving the Mixture Estimation problem. We show it can actually be formulated as a constrained convex program. We observe that the objective function only depends on Q through the corresponding mixture likelihood vector $\mathbf{g}(Q)$. Therefore, instead of searching over the mixing

distributions, we directly search over the mixture likelihood vectors, obtaining the following equivalence:

$$\min_{Q\in\mathcal{Q}} \mathsf{loss}(\mathbf{g}(Q)) \quad \equiv \quad \min_{\mathbf{g}\in\{\mathbf{g}(Q)\colon Q\in\mathcal{Q}\}} \mathsf{loss}(\mathbf{g}),$$

where we have dropped the explicit dependence of loss on Data for simplicity of notation.

With the above equivalence, our ability to solve the Mixture Estimation problem depends on our ability to describe the constraint set $\{\mathbf{g}(Q)\colon Q\in\mathcal{Q}\}$. We show next that this constraint set can indeed be expressed as a convex set. For that, analogous to the mixture likelihood vector above, define the *atomic likelihood vector* $\boldsymbol{f}(\boldsymbol{\omega}) \stackrel{\text{def}}{=} (f_{jt}(\boldsymbol{\omega})\colon t\in[T], j\in S_t)$. We let $\mathcal{P}=\{\boldsymbol{f}(\boldsymbol{\omega})\colon \boldsymbol{\omega}\in\mathbb{R}^D\}$ denote the set of all possible atomic likelihood vectors and $\overline{\mathcal{P}}$ denote its closure.[3] It is clear that if $Q\in\mathcal{Q}$ is a discrete distribution with finite support $\boldsymbol{\omega}_1,\boldsymbol{\omega}_2,\ldots,\boldsymbol{\omega}_K$ and corresponding mixing weights $\alpha_1,\alpha_2,\ldots,\alpha_K$, then $\mathbf{g}(Q)=\sum_{k=1}^K \alpha_k \boldsymbol{f}(\boldsymbol{\omega}_k)$, so $\mathbf{g}(Q)$ belongs to the convex hull, $\mathrm{conv}(\overline{\mathcal{P}})$, of vectors in $\overline{\mathcal{P}}$, defined as

$$\mathrm{conv}(\overline{\mathcal{P}}) = \{\sum_{\boldsymbol{f}\in\mathcal{F}} \alpha_{\boldsymbol{f}}\boldsymbol{f} \;:\; \mathcal{F}\subset\overline{\mathcal{P}} \text{ is finite and } \sum_{\boldsymbol{f}\in\mathcal{F}}\alpha_{\boldsymbol{f}}=1, \alpha_{\boldsymbol{f}}\geq 0 \;\; \forall \boldsymbol{f}\in\mathcal{F}\}.$$

It can be verified that $\mathrm{conv}(\overline{\mathcal{P}})$ is a convex set in $\mathbb{R}^M$. In other words, for any discrete mixing distribution Q with finite support, we can express $\mathbf{g}(Q)$ as a convex combination of atomic likelihood vectors $\{\boldsymbol{f}\}_{\boldsymbol{f}\in\mathcal{F}}$ for some finite subset $\mathcal{F}\subset\overline{\mathcal{P}}$. More generally, it can be shown [Lin83] that $\{\mathbf{g}(Q)\colon Q\in\mathcal{Q}\}=\mathrm{conv}(\overline{\mathcal{P}})$ when $\overline{\mathcal{P}}$ is compact (which we establish in Lemma 3.1 below).

Thus, instead of solving Mixture Estimation, we can equivalently solve the following problem:

$$\min_{\mathbf{g}\in\mathrm{conv}(\overline{\mathcal{P}})} \mathsf{loss}(\mathbf{g}) \qquad \text{(Convex Mixture)}$$

We now show that the above is a constrained convex program.

Lemma 3.1. *For any convex function $\mathsf{loss}(\cdot)\colon[0,1]^M\to\mathbb{R}_+$, Convex Mixture is a convex program with a compact constraint set in the Euclidean space.*

Proof. The objective function is by definition convex. Therefore, it is sufficient to show that the constraint set $\mathrm{conv}(\overline{\mathcal{P}})$ is convex. For that, we note that since all entries of $\boldsymbol{f}(\boldsymbol{\omega})$ are between 0 and 1 for any $\boldsymbol{\omega}\in\mathbb{R}^D$, all limit points of the set $\{\boldsymbol{f}(\boldsymbol{\omega})\colon \boldsymbol{\omega}\in\mathbb{R}^D\}$ are also bounded. Therefore, $\overline{\mathcal{P}}$ is a bounded set in $\mathbb{R}^M$. As it is closed by definition, it follows from the Heine-Borel theorem that $\overline{\mathcal{P}}$ is compact. Further, since the convex hull of a compact subset of the Euclidean space is compact, it follows that $\mathrm{conv}(\overline{\mathcal{P}})$ is compact, and by the definition of a convex hull it is convex. The claim then follows. □

So, our task now is to solve the Convex Mixture problem. However, solving it alone does not provide the mixing distribution—it only provides the optimal mixture likelihood vector. We show next that the conditional gradient algorithm is the ideal candidate to not only obtain the optimal mixture likelihood vector, but also the optimal mixing distribution.

[3] For technical reasons, as will become clear in Section 3.4.2, we need to consider the closure of the set $\mathcal{P}$, which also contains all limit points of convergent sequences in the set $\mathcal{P}$.

3.3 Conditional gradient algorithm for estimating the mixing distribution

We now apply the conditional gradient (hereafter CG) algorithm to solve the CONVEX MIXTURE problem. The CG algorithm [FW56, Jag13] is an iterative method for solving constrained convex programs. It has seen an impressive revival in the machine learning literature recently because of its favorable properties compared to standard projected/proximal gradient methods, such as efficient handling of complex constraint set (see [Jag11] for an excellent overview). We describe here how it applies to solving $\min_{\mathbf{g}\in\text{conv}(\overline{\mathcal{P}})} \mathsf{loss}(\mathbf{g})$.

The CG algorithm is an iterative first-order method that starts from an initial feasible solution, say $\mathbf{g}^{(0)} \in \text{conv}(\overline{\mathcal{P}})$, and generates a sequence of feasible solutions $\mathbf{g}^{(1)}, \mathbf{g}^{(2)}, \ldots$ that converge to the optimal solution. In each iteration $k \geq 1$, it computes a *descent direction* $\mathbf{d}$ such that $\left\langle \mathbf{d}, \nabla\mathsf{loss}(\mathbf{g}^{(k-1)}) \right\rangle < 0$ and takes a suitable step in that direction (see for instance [NW06]). The main distinction of the CG algorithm is that it always chooses *feasible* descent steps, where by a feasible step we mean a step from the current solution towards the next solution such that the next solution remains feasible as long as the current is feasible. By contrast, other classical algorithms may take infeasible steps, which are then projected back onto the feasible region after each step; such projection steps are usually computationally expensive. To find a feasible step, the CG algorithm first obtains a descent direction by optimizing a linear approximation of the convex loss function at the current iterate $\mathbf{g}^{(k-1)}$:

$$\min_{\mathbf{v}\in\text{conv}(\overline{\mathcal{P}})} \left\langle \nabla\mathsf{loss}(\mathbf{g}^{(k-1)}), \mathbf{v} - \mathbf{g}^{(k-1)} \right\rangle,$$

where the objective function in the above subproblem describes a supporting hyperplane to the convex loss function $\mathsf{loss}(\cdot)$ at the current iterate $\mathbf{g}^{(k-1)}$. The optimal solution, say, $\mathbf{v}^*$, provides the optimal direction $\mathbf{d}^* = \mathbf{v}^* - \mathbf{g}^{(k-1)}$. It can be shown that $\mathbf{d}^*$ is a descent direction if $\mathbf{g}^{(k-1)}$ is not already an optimal solution to the CONVEX MIXTURE problem. The next solution $\mathbf{g}^{(k)}$ is obtained by taking a step $\alpha \in [0, 1]$ in the direction of $\mathbf{d}^*$, so that $\mathbf{g}^{(k)} = \mathbf{g}^{(k-1)} + \alpha\mathbf{d}^* = \alpha\mathbf{v}^* + (1-\alpha)\mathbf{g}^{(k-1)}$. Since $\mathbf{v}^*$ and $\mathbf{g}^{(k-1)}$ both belong to $\text{conv}(\overline{\mathcal{P}})$ and $\text{conv}(\overline{\mathcal{P}})$ is convex, it follows that $\mathbf{g}^{(k)} \in \text{conv}(\overline{\mathcal{P}})$ for any $\alpha \in [0, 1]$.

Solving the above subproblem is the most computationally challenging component in each iteration of the CG algorithm. In our context, we have additional structure that we can exploit to solve this subproblem. Specifically, the objective function is linear in the decision variable $\mathbf{v}$. And, linear functions always achieve optimal solutions at extreme points when optimized over a convex set. Our constraint set $\text{conv}(\overline{\mathcal{P}})$ is the convex hull of all the atomic likelihood vectors in $\overline{\mathcal{P}}$. Therefore, the set of extreme points of the constraint set is a subset of $\overline{\mathcal{P}}$. It thus follows that it is sufficient to search over the set of all atomic likelihood vectors in $\overline{\mathcal{P}}$, resulting in the following optimization problem:

$$\min_{\boldsymbol{f}\in\overline{\mathcal{P}}} \left\langle \nabla\mathsf{loss}(\mathbf{g}^{(k-1)}), \boldsymbol{f} - \mathbf{g}^{(k-1)} \right\rangle \qquad \text{(support finding step)}$$

Our ability to solve the support finding step efficiently depends on the structure of the set $\overline{\mathcal{P}}$. We discuss this aspect in our empirical studies (Sections 3.6.1 and 3.6.2). For now, suppose

that we have access to an oracle that returns an optimal solution, say, $\boldsymbol{f}^{(k)}$, to the support finding step in each iteration k.

In summary, in each iteration, the CG algorithm finds a new customer type (or atomic likelihood vector) $\boldsymbol{f}^{(k)} \in \overline{\mathcal{P}}$ and obtains the new solution $\mathbf{g}^{(k)} = \alpha \boldsymbol{f}^{(k)} + (1-\alpha)\mathbf{g}^{(k-1)}$ by putting a probability mass α on the new customer type $\boldsymbol{f}^{(k)}$ and the remaining probability mass $1-\alpha$ on the previous solution $\mathbf{g}^{(k-1)}$, for some $\alpha \in [0,1]$. In other words, the CG algorithm is iteratively adding customer types $\boldsymbol{f}^{(1)}, \boldsymbol{f}^{(2)}, \ldots$ to the support of the mixing distribution. This aspect of the CG algorithm makes it most attractive for estimating mixture distributions, and is the reason we refer to the subproblem in each iteration as the support finding step. In particular, it has two implications: (a) by maintaining the individual customer types and the step sizes, we can maintain the entire mixing distribution along with the current solution $\mathbf{g}^{(k)}$, in each iteration k (see below for details); and (b) since each iteration adds (at most) one new customer type to the support, terminating the program at iteration K results in a distribution with at most K mixture components. We use the latter property to control the complexity, as measured in terms of the number of mixture components, of the recovered mixing distribution (see the discussion below on "Stopping conditions").

The standard variant of the CG algorithm does a line-search to compute the optimal step size that results in the maximum improvement in the objective value. Instead, we use the "fully corrective" Frank-Wolfe (FCFW) variant [SSSZ10] which after finding $\boldsymbol{f}^{(k)}$ at iteration k, re-optimizes the loss function $\mathsf{loss}(\mathbf{g})$ over the convex hull of the initial solution $\mathbf{g}^{(0)}$ and the atomic likelihood vectors $\boldsymbol{f}^{(1)}, \boldsymbol{f}^{(2)}, \ldots, \boldsymbol{f}^{(k)}$ found so far. More precisely, the algorithm computes weights $\boldsymbol{\alpha}^{(k)}$ from the $(k+1)$-dimensional simplex Δ^k that minimize the loss function and obtains the next iterate $\mathbf{g}^{(k)} := \alpha_0^{(k)}\mathbf{g}^{(0)} + \sum_{s=1}^{k} \alpha_s^{(k)} \boldsymbol{f}^{(s)}$. The weights $\boldsymbol{\alpha}^{(k)} = (\alpha_0^{(k)}, \alpha_1^{(k)}, \alpha_2^{(k)}, \ldots, \alpha_k^{(k)})$ represent the proportions of each of the mixture components. This variant of the CG algorithm makes more progress in each iteration and is therefore most suited when the subproblems in the support finding step are hard to solve. It also promotes sparser solutions [Jag13] containing fewer mixture components. Algorithm 8 summarizes the above procedure.

Algorithm 8 CG algorithm for estimating the mixing distribution

Initialize: $k = 0$; $\mathbf{g}^{(0)} \in \overline{\mathcal{P}}$ such that $\mathsf{loss}(\mathbf{g}^{(0)})$ is bounded above and $\boldsymbol{\alpha}^{(0)} = (1)$

while stopping condition is not met **do**

  $k \leftarrow k + 1$

  Compute $\boldsymbol{f}^{(k)} \in \arg\min_{\boldsymbol{f} \in \overline{\mathcal{P}}} \left\langle \nabla \mathsf{loss}(\mathbf{g}^{(k-1)}), \boldsymbol{f} \right\rangle$ (support finding step)

  Compute $\boldsymbol{\alpha}^{(k)} \in \arg\min_{\boldsymbol{\alpha} \in \Delta^k} \mathsf{loss}\left(\alpha_0 \mathbf{g}^{(0)} + \sum_{s=1}^{k} \alpha_s \boldsymbol{f}^{(s)}\right)$ (proportions update step)

  Update $\mathbf{g}^{(k)} := \alpha_0^{(k)} \mathbf{g}^{(0)} + \sum_{s=1}^{k} \alpha_s^{(k)} \boldsymbol{f}^{(s)}$

end while

Output: mixture proportions $\alpha_0^{(k)}, \alpha_1^{(k)}, \alpha_2^{(k)}, \ldots, \alpha_k^{(k)}$ and customer types $\mathbf{g}^{(0)}, \boldsymbol{f}^{(1)}, \boldsymbol{f}^{(2)}, \ldots, \boldsymbol{f}^{(k)}$

We make a few remarks about the algorithm. First, the algorithm outputs both the support and the mixture proportions of the mixing distribution, as desired. Second, the proportions update step is also a constrained convex optimization problem, but over a much smaller domain compared to $\mathrm{conv}(\overline{\mathcal{P}})$. We show below that this step can be solved efficiently. Finally, Algorithm 8 is agnostic to the choice of the loss function loss (so long as it is convex and differentiable) and readily applies to both the NLL and SQ loss functions.

3.3.1 Implementation Details

Here, we discuss a few key implementation details for Algorithm 8.

Solving the proportions update step. This step is itself a constrained convex program, so we use the CG algorithm to solve it. Instead of the variant described above, we adopt the approach of [KLJS15] who recently proposed a modified Frank-Wolfe algorithm to approximately solve the proportions update step. In contrast to the standard CG algorithm described above, this variant performs two kinds of steps to update the support of the mixing distribution in each iteration: a support finding step that finds a customer type to be added to the mixture and an "away" step [GM86] that reduces probability mass (possibly to zero) from a customer type in the existing mixture distribution. Moreover, the support finding step can be solved exactly by searching over the $k+1$ extreme points of the $(k+1)$-dimensional simplex Δ^k. The next iterate is then computed based on which step—support finding step or away step—results in higher improvement in the objective value (see Alg. 3 in Appendix B of [KLJS15] for the details). The presence of away steps means that we can (sometimes) 'drop' existing customer types from the mixing distribution, thereby resulting in solutions with fewer number of mixture components.

Initialization. We can start with any $\mathbf{g}^{(0)} \in \overline{\mathcal{P}}$ as the initial solution. The only caveat is that for the NLL loss function, we need to ensure that $\mathbf{g}^{(0)} > \mathbf{0}$ so that the gradient

$\nabla\mathsf{NLL}(\mathbf{g}^{(0)})$ is well-defined. Since we are fitting a mixture of logit models, a natural choice is to fit an LC-MNL model with a "small" number of classes (or even a single class MNL model) to the data and use that as the initialization. In particular, the MNL log-likelihood objective is globally concave in the parameter $\boldsymbol{\omega}$ and there exists efficient algorithms [Hun04] for its estimation that converge quickly in practice. In our empirical case studies, we initialize by fitting a 2-class LC-MNL model to the data.

Stopping conditions. We can use many stopping conditions to terminate the algorithm: (1) [Jag13] showed that if the subproblem can be solved optimally in each iteration, then we can compute an upper bound on the "optimality gap" of the current solution $\mathbf{g}^{(k)}$, i.e. $\mathsf{loss}(\mathbf{g}^{(k)}) - \mathsf{loss}(\mathbf{g}^*)$ where $\mathbf{g}^*$ denotes the optimal solution to the Convex Mixture problem. In this case, we can choose an arbitrarily small $\delta > 0$ and choose to terminate the algorithm when $\mathsf{loss}(\mathbf{g}^{(k)}) - \mathsf{loss}(\mathbf{g}^*) \leq \delta$. However, this might result in overfitting—because of the presence of a large number of mixture components—and consequently, perform poorly in out-of-sample predictions. (2) We can utilize standard information-theoretic measures proposed in the mixture modeling literature [MP00] such as Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC) etc. that capture model complexity as a function of the number of mixture components and prevent overfitting. (3) Finally, a simple way to control for model complexity is to just cap the number of iterations of the algorithm at some $k = K_{\max}$ according to the maximum number of customer types that we may be interested in finding. This ensures that the estimated mixture is composed of at most $K_{\max}$ types and we use this stopping condition in our empirical case studies.

3.4 Theoretical analysis of the estimator

In this section, we derive the convergence rate of our estimator and also theoretically characterize the customer types recovered by our method.

3.4.1 Convergence rate of the estimator

To state our result on the convergence rate, we need the following notation. For each offer-set S_t, let $\boldsymbol{y}_t \stackrel{\text{def}}{=} (y_{jt})_{j \in S_t}$ denote the vector of sales fractions for offer-set S_t. Let $H(\boldsymbol{y}_t) \stackrel{\text{def}}{=} -\sum_{j \in S_t} y_{jt} \log y_{jt}$ denote the entropy of the vector $\boldsymbol{y}_t$. As $0 \leq y_{jt} \leq 1$, $H(\boldsymbol{y}_t) \geq 0$ for all $t \in [T]$.[4] Moreover, let $D_{KL}(p\|q) \stackrel{\text{def}}{=} p\log(p/q) + (1-p)\log((1-p)/(1-q))$ denote the relative entropy (aka KL-divergence) between p and q for any $0 \leq p, q \leq 1$. It is a known fact that $D_{KL}(p\|q) \geq 0$ for all $0 \leq p, q \leq 1$ and $D_{KL}(p\|q) = 0$ if and only if $p = q$. Finally, let $y_{t,\min} \stackrel{\text{def}}{=} \min\{y_{jt} \mid j \in S_t \text{ s.t. } y_{jt} > 0\}$ and $y_{\min} \stackrel{\text{def}}{=} \min\{y_{t,\min} \mid t \in [T]\}$. Then, we can establish the following convergence guarantee:

Theorem 3.2 (Sublinear convergence)**.** *Let $\mathbf{g}^*$ denote the optimal solution to the* Convex Mixture *problem, and $\mathbf{g}^{(k)}$ denote the kth iterate generated by Algorithm 8. Then, for the*

[4] We use the standard convention that $0 \cdot \log 0 = 0$ when computing the entropy.

loss functions defined in Section 3.2.2, it follows

$$\mathsf{SQ}(\mathbf{g}^{(k)}) - \mathsf{SQ}(\mathbf{g}^*) \leq \frac{4}{k+2} \qquad \textit{for all } k \geq 1,$$

$$\mathsf{NLL}(\mathbf{g}^{(k)}) - \mathsf{NLL}(\mathbf{g}^*) \leq \frac{4}{\xi_{\min}^2 \cdot (k+\delta)} \qquad \textit{for all } k \geq K \textit{ for some constant } \delta \textit{ and index } K,$$

where $\xi_{\min}$ *is the smallest* ξ *such that* $0 < \xi \leq y_{\min}$ *and*

$$\min_{1 \leq t \leq T} N_t \cdot D_{KL}(y_{t,\min} \| \xi_{\min}) \leq N \cdot \mathsf{NLL}(\mathbf{g}^{(0)}) - \sum_{t=1}^{T} N_t \cdot H(\boldsymbol{y}_t).$$

Such a $\xi_{\min}$ *always exists.*

The result above establishes an $O(1/k)$ convergence guarantee for our estimator for both loss functions, assuming that the support finding step in Algorithm 8 can be solved optimally. The detailed proof is given in Appendix C.1; here, we provide a proof sketch. Our proof builds on existing techniques developed for establishing convergence rates of the CG algorithm. This is an active area of research with different rates having been derived for different variants of the CG algorithm, under different assumptions for the structures of the objective function and the constraint set [Jag13, GH15, LJJ15]. The convergence guarantee for the squared loss function, in fact, follows directly from the existing result in [Jag13], which shows that the CG algorithm converges at an $O(1/k)$ rate if the so-called *curvature constant* is bounded from above. If the domain is bounded and the hessian of the objective function is bounded from above, then the curvature constant is known to be bounded from above. In our case, the domain $\text{conv}(\overline{\mathcal{P}})$ is bounded (since any vector $\boldsymbol{f} \in \text{conv}(\overline{\mathcal{P}})$ has entries between 0 and 1). The hessian of the squared loss is a diagonal matrix, where each entry is bounded above by 1. Therefore, it follows that the curvature constant is bounded from above, thus allowing us to establish the $O(1/k)$ guarantee by directly invoking existing results.

The hessian of the NLL loss function, on the other hand, is not bounded from above. It is a diagonal matrix with the entry corresponding to (product, offer-set) pair (j, S_t) equal to y_{jt}/g_{jt}^2. Since g_{jt} can be arbitrarily close to 0 in the domain $\text{conv}(\overline{\mathcal{P}})$, the diagonal entries are not bounded from above, and thus, existing results don't directly apply. To address this issue, suppose that we can establish a non-trivial lower bound, say, $\xi^* > 0$, for the optimal solution $\mathbf{g}^*$ so that $\mathrm{g}^*_{jt} \geq \xi^* > 0$ for all $t \in [T]$ and all $j \in S_t$. It then follows that the hessian of the NLL loss function is bounded from above when the domain is restricted to $\tilde{\mathcal{D}} \stackrel{\text{def}}{=} \{\mathbf{g} \in \text{conv}(\overline{\mathcal{P}}) : \mathrm{g}_{jt} \geq \xi^* \; \forall \, t \in [T]; \forall \, j \in S_t\}$. And, if we solve Convex Mixture over the restricted domain $\tilde{\mathcal{D}}$,[5] we immediately obtain the $O(1/k)$ convergence rate.

While solving Convex Mixture over the restricted domain $\tilde{\mathcal{D}}$ is feasible in principle, it is difficult to implement in practice because computing a good lower bound ξ^* may not be straightforward. Instead, we show that running the fully corrective variant of the CG algorithm (the variant implemented in Algorithm 8), while being agnostic to a lower bound,

[5] It can be verified that the constraint set $\tilde{\mathcal{D}}$ is still compact and convex.

still converges at $O(1/k)$ rate. For that, we first show that each iterate $\mathbf{g}^{(k)}$ generated by Algorithm 8 is bounded from below by $\xi_{\min}$, where $\xi_{\min}$ is as defined in Theorem 3.2. Then, we exploit this property to establish the $O(1/k)$ convergence rate with the constant scaling in $1/\xi_{\min}^2$.

To get the best convergence rate, we need to use the tightest lower bound $\xi_{\min}$. Our bound is derived for general cases, and in this generality, the bound is tight. To see that, consider the setting when the observations consist of only market shares, so that $T = 1$, $S_1 = [n]$, and the sales fractions $\boldsymbol{y}_1$ comprise the observed market shares. In this case, it can be shown that the optimal solution $\mathbf{g}^* = \boldsymbol{y}_1$.[6] When Algorithm 8 is initialized at $\mathbf{g}^{(0)} = \boldsymbol{y}_1$, it follows from the definition that $\xi_{\min} = y_{\min} = y_{1,\min}$, which is the tightest bound possible.

We can also derive a simple-to-compute (lower) bound for $\xi_{\min}$, as stated in the following proposition:

Proposition 3.2.1. *Let* $N_{\min} = \min\{N_{jt} \mid t \in [T]; j \in S_t \text{ s.t. } N_{jt} > 0\}$. *Then, it follows that*

$$y_{\min} \geq \xi_{\min} \geq y_{\min} \cdot \exp\left(-1 - \frac{N \cdot \mathsf{NLL}(\mathbf{g}^{(0)}) - \sum_{t=1}^{T} N_t \cdot H(\boldsymbol{y}_t)}{N_{\min}}\right).$$

When $T = 1$, $S_1 = [n]$, and $\mathbf{g}^{(0)} = \boldsymbol{y}_1$, it follows from the above proposition that $y_{\min} \geq \xi_{\min} \geq y_{\min}/e$. Therefore, the simple-to-compute (lower) bound loses a factor of e in this case.

Remark. Theorem 3.2 assumes that the support finding step in Algorithm 8 can be solved optimally.[7] In cases where the optimal solution cannot be found, a weaker convergence guarantee can be established as long as the iterates are (sufficiently) improving, i.e., $\mathsf{loss}(\mathbf{g}^{(k)}) < \mathsf{loss}(\mathbf{g}^{(k-1)})$, for each iteration k. In this case, it follows from existing results (see for instance [Zan69]) that the sequence of iterates converges to a stationary point, which in the case of a convex program is an optimal solution.

3.4.2 Characterization of the recovered mixture types

We now focus on the support finding step and characterize the structure of the optimal solution. These solutions comprise the support of the resulting mixture distribution. In each iteration k, the support finding step involves solving the following problem:

$$\min_{\boldsymbol{f} \in \overline{\mathcal{P}}} \sum_{t=1}^{T} \sum_{j \in S_t} c_{jt}^{(k)} f_{jt},$$

where $c_{jt}^{(k)} = \left(\nabla \mathsf{loss}(\mathbf{g}^{(k-1)})\right)_{jt}$. The optimal solution $\boldsymbol{f}^{(k)}$ to the above problem lies either in $\mathcal{P}$ or $\overline{\mathcal{P}} \setminus \mathcal{P}$. If it lies in $\mathcal{P}$, then (by definition) there exists a parameter vector $\boldsymbol{\omega}_k \in \mathbb{R}^D$

[6] Provided the product features satisfy certain structural conditions; see Theorem 3.5.

[7] Actually, [Jag13] showed that solving it approximately with some fixed additive error is also sufficient to ensure the $O(1/k)$ convergence rate.

such that $\boldsymbol{f}(\boldsymbol{\omega}_k) = \boldsymbol{f}^{(k)}$, so that any such $\boldsymbol{\omega}_k$ may be used to describe the customer type and make the choice probability prediction $e^{\boldsymbol{\omega}_k^\top \mathbf{z}_{jS}} / \left(\sum_{\ell \in S} e^{\boldsymbol{\omega}_k^\top \mathbf{z}_{\ell S}}\right)$ for the probability of choosing product j from some offer-set S. However, if the optimal solution $\boldsymbol{f}^{(k)}$ lies in the boundary, i.e. $\overline{\mathcal{P}} \setminus \mathcal{P}$, then there is no straightforward way to characterize the customer type or make out-of-sample predictions. To deal with this challenge, we provide a compact characterization of what we call the *boundary* types, defined as follows:

Definition 3.1 (Boundary and Non-boundary types)**.** A customer type $\boldsymbol{f}$ is called a boundary type if $\boldsymbol{f} \in \overline{\mathcal{P}} \setminus \mathcal{P}$, and a non-boundary type, otherwise.

We show below that each boundary type is characterized by two parameters $(\boldsymbol{\omega}_0, \boldsymbol{\theta})$:

Theorem 3.3 (Characterization of boundary types)**.** *Given a boundary type $\boldsymbol{f}$ in $\overline{\mathcal{P}} \setminus \mathcal{P}$, there exist parameters $\boldsymbol{\omega}_0, \boldsymbol{\theta} \in \mathbb{R}^D$ such that, for each $1 \le t \le T$ and $j \in S_t$, we have*

$$f_{jt} = \lim_{r \to \infty} \frac{\exp\left((\boldsymbol{\omega}_0 + r \cdot \boldsymbol{\theta})^\top \mathbf{z}_{jt}\right)}{\sum_{\ell \in S_t} \exp\left((\boldsymbol{\omega}_0 + r \cdot \boldsymbol{\theta})^\top \mathbf{z}_{\ell t}\right)}.$$

Furthermore, $f_{jt} = 0$ for at least one (product, offer-set) pair (j, S_t).

The proof in Appendix C.2 shows how to compute the parameters $(\boldsymbol{\omega}_0, \boldsymbol{\theta})$ given any boundary type $\boldsymbol{f} \in \overline{\mathcal{P}} \setminus \mathcal{P}$. Here, we focus on understanding the implications of the above characterization.

The key aspect of our characterization is a preference ordering over the products defined by the parameter vector $\boldsymbol{\theta}$. This preference order determines the choice of the products from a given offer-set. For ease of exposition, we describe the preference ordering for the case when product features don't change with the offer-set, so we write $\mathbf{z}_j$ instead of $\mathbf{z}_{jt}$ for the feature vector of product j. The discussion below extends immediately to the more general case by associating a separate product to each feature vector of interest. To describe the preference order, define product utility $u_j \stackrel{\text{def}}{=} \boldsymbol{\theta}^\top \mathbf{z}_j$ for product j. These utility values can be visualized as projections of the product feature vectors $\mathbf{z}_j$'s onto the vector $\boldsymbol{\theta}$. They define a preference order $\succeq$ among the products such that $j \succeq j'$, read as "product j is weakly preferred over product j'," if and only if $u_j \ge u_{j'}$. The relation $\succeq$ is in general a weak ordering and *not* a strict ordering because product utilities may be equal. In order to explicitly capture indifferences, we write $j \succ j'$ if $u_j > u_{j'}$ and $j \sim j'$ if $u_j = u_{j'}$.

Now, when offered a set S, customers of this type purchase only the most preferred products as determined according to the preference order $\succeq$. To see that, let $C(S)$ denote the set of most preferred products in S, so that for all $j \in C(S)$, we have $j \sim \ell$ if $\ell \in C(S)$ and $j \succ \ell$ if $\ell \in S \setminus C(S)$. Let $u^* \stackrel{\text{def}}{=} \max\{u_j : j \in S\}$ denote the maximum utility among the products in S. We have that $u^* = u_j$ for all $j \in C(S)$ and $u^* > u_j$ for all $j \in S \setminus C(S)$. Given this and multiplying the numerator and denominator of the choice probabilities defined

in Theorem 3.3 by $e^{-r\cdot u^*}$, we can write for any product $j \in S$,

$$\frac{\exp\left((\boldsymbol{\omega}_0 + r\cdot\boldsymbol{\theta})^\top \mathbf{z}_j\right)}{\sum_{\ell\in S}\exp\left((\boldsymbol{\omega}_0 + r\cdot\boldsymbol{\theta})^\top \mathbf{z}_\ell\right)} = \frac{e^{-r\cdot(u^*-u_j)}\cdot\exp(\boldsymbol{\omega}_0^\top \mathbf{z}_j)}{\sum_{\ell\in C(S)}\exp(\boldsymbol{\omega}_0^\top \mathbf{z}_\ell) + \sum_{\ell\in S\setminus C(S)} e^{-r\cdot(u^*-u_\ell)}\cdot\exp(\boldsymbol{\omega}_0^\top \mathbf{z}_\ell)}. \tag{3.3}$$

When we take the limit as $r\to\infty$, each of the terms $e^{-r\cdot(u_\ell-u^*)}$, $\ell\in S\setminus C(S)$, goes to zero, so the denominator converges to $\sum_{\ell\in C(S)}\exp(\boldsymbol{\omega}_0^\top \mathbf{z}_\ell)$. The numerator converges to $\exp(\boldsymbol{\omega}_0^\top \mathbf{z}_j)$ if $j\in C(S)$ and 0 if $j\in S\setminus C(S)$. Therefore, we obtain the following choice probability prediction $f_{j,S}(\boldsymbol{\omega}_0,\boldsymbol{\theta})$ for any product j and offer-set S from Theorem 3.3:

$$f_{j,S}(\boldsymbol{\omega}_0,\boldsymbol{\theta}) = \begin{cases} \exp(\boldsymbol{\omega}_0^\top \mathbf{z}_j)/\left(\sum_{\ell\in C(S)}\exp(\boldsymbol{\omega}_0^\top \mathbf{z}_\ell)\right), & \text{if } j\in C(S) \text{ and} \\ 0, & \text{if } j\in S\setminus C(S). \end{cases}$$

From the discussion above, we note the contrasting roles of the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\omega}_0$. The parameter vector $\boldsymbol{\theta}$ (through the preference ordering $\succeq$ it induces) determines the *consideration set* $C(S)$, whereas the parameter vector $\boldsymbol{\omega}_0$ determines the logit choice probabilities from within the consideration set. The parameter vector $\boldsymbol{\theta}$ dictates how a product's features impact its inclusion into the consideration set. For instance, suppose u^* is the maximum utility in offer-set S and product j with utility $u_j < u^*$ is not in consideration now. Further, suppose one of the features is price and the corresponding coefficient is $\theta_p < 0$. Then, product j will enter into consideration only if its price is sufficiently dropped to make its utility greater than or equal to u^*. In other words, the price should be dropped by at least $(u^*-u_j)/|\theta_p|$ to ensure consideration of product j.

The choice behavior we identify for the boundary types is consistent with existing literature, which establishes that customers often consider a subset of the products on offer before making the choice [JR16]. In fact, consideration sets of the kind we identify are a special case of the linear compensatory decision rule that has been used as a heuristic for forming consideration sets in existing literature [Hau14]. The rule computes the utility for each product as a weighted sum of the feature values and chooses all products that have a utility greater than a cutoff to be part of the consideration set. Finally, multiple distinct tuples of parameters $(\boldsymbol{\omega}_0,\boldsymbol{\theta})$ can result in the same limiting choice probabilities $\boldsymbol{f}$ for the observed data. Since the data do not provide any further guidance, we arbitrarily select one of them. Studying the impact of different selection rules on the prediction accuracy is a promising avenue for future work.

We conclude this subsection with the following systematic procedure that summarizes our discussion for making choice predictions for a boundary type, on new product features and/or offer-set combinations:

Algorithm 9 Predicting choice probabilities for boundary type $\boldsymbol{f}(\boldsymbol{\omega}_0, \boldsymbol{\theta})$

1: **Input:** Offer-set $S \subseteq [n]$ with product features $\mathbf{z}_{jS} \in \mathbb{R}^D$ for each $j \in S$

2: Compute utilities $u_j = \boldsymbol{\theta}^\top \mathbf{z}_{jS}$ for each $j \in S$.

3: Form consideration set $C(S) = \{j \in S \mid u_j = \max_{\ell \in S} u_\ell\}$

4: For any $j \notin C(S)$, set $f_{j,S}(\boldsymbol{\omega}_0, \boldsymbol{\theta}) \leftarrow 0$

5: For any $j \in C(S)$, set

$$f_{j,S}(\boldsymbol{\omega}_0, \boldsymbol{\theta}) \leftarrow \frac{\exp\left(\boldsymbol{\omega}_0^\top \mathbf{z}_{jS}\right)}{\sum_{\ell \in C(S)} \exp\left(\boldsymbol{\omega}_0^\top \mathbf{z}_{\ell S}\right)}$$

6: **Output:** Choice probabilities $\{f_{j,S}(\boldsymbol{\omega}_0, \boldsymbol{\theta}) : j \in S\}$

3.4.3 Analysis of recovered distribution for two special cases

We now analyze scenarios under which the optimal solution to the support finding step is indeed a boundary type. This helps in providing further insights into the structure of the recovered mixing distribution. Solving the support finding step in the general case is a hard problem and therefore, to keep the analysis tractable, we focus on the setting in which the data consist of sales counts in a single time-period when all products are offered (such as market shares data). For this case, the notation can be simplified.

Since there is only a single offer-set $S_1 = [n]$, we represent the features as $\mathbf{z}_i$ for each product $i \in [n]$. Further, the sales counts can be represented using a single vector $\boldsymbol{y} := (y_1, y_2, \ldots, y_n) \in [0,1]^n$ such that $\sum_{i=1}^n y_i = 1$ where $y_i \geq 0$ is the fraction of sales for product i. The choice probabilities $\boldsymbol{f} \in \overline{\mathcal{P}}$ are of the form $\boldsymbol{f} = (f_1, f_2, \ldots, f_n)$, also satisfying $\sum_{i=1}^n f_i = 1$. Similarly, the estimates produced by Algorithm 8 at any iteration k are of the form $\mathbf{g}^{(k)} = (\mathrm{g}_1^{(k)}, \mathrm{g}_2^{(k)}, \ldots, \mathrm{g}_n^{(k)})$, where again $\sum_{i=1}^n \mathrm{g}_i^{(k)} = 1$. With this notation, the loss functions defined in Section 3.2.2 can be written as:

$$\mathsf{NLL}(\mathbf{g}) = -\sum_{i=1}^n y_i \log\left(\mathrm{g}_i\right) \quad ; \quad \mathsf{SQ}(\mathbf{g}) = \frac{1}{2}\sum_{i=1}^n \left(y_i - \mathrm{g}_i\right)^2, \tag{3.4}$$

while the support finding step is of the form, with $c_i \stackrel{\text{def}}{=} -\left(\nabla \mathsf{loss}(\mathbf{g}^{(k-1)})\right)_i$ for each $i \in [n]$ (we switch to maximization to aid the analysis below):

$$\max_{\boldsymbol{f} \in \overline{\mathcal{P}}} \sum_{i=1}^n c_i \cdot f_i, \tag{3.5}$$

where we drop the explicit dependence of the coefficient c_i on the iteration number k for simplicity of notation. We analyze the optimal solution to the above subproblem under two

cases: (1) all product features are continuous, and (2) some product features are binary.[8]

All product features are continuous. When all features are continuous, the optimal solution to subproblem (3.5) depends on the geometric structure of the observed product features. Specifically, we consider the (convex) polytope formed by the convex hull of the product features $\mathbf{z}_1, \ldots, \mathbf{z}_n$, denoted as $\mathcal{Z}_n \stackrel{\text{def}}{=} \text{conv}(\{\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n\})$. For this polytope, we define an extreme point as:

Definition 3.2 (Extreme points)**.** $\mathbf{z}_j$ is called an *extreme point* of the convex polytope $\mathcal{Z}_n$ if $\mathbf{z}_j \notin \text{conv}(\{\mathbf{z}_i \ : \ i \neq j\})$. Equivalently, extreme points correspond to vertices of $\mathcal{Z}_n$.

With this definition, we can establish conditions under which a boundary type is an optimal solution to the support finding step (3.5). In particular, we have the following result:

Theorem 3.4 (Recovery of boundary types)**.** *Let* $j^{\max} = \arg\max_{j \in [n]} c_j$. *If* $\mathbf{z}_{j^{\max}}$ *is an extreme point, then the boundary type* $\boldsymbol{f}(\mathbf{0}, \boldsymbol{\theta}_{j^{\max}})$ *is an optimal solution to* support finding step (3.5)*, where* $\boldsymbol{\theta}_{j^{\max}}$ *is such that* $\boldsymbol{\theta}_{j^{\max}}^\top \mathbf{z}_{j^{\max}} > \boldsymbol{\theta}_{j^{\max}}^\top \mathbf{z}_j$ *for all* $j \neq j^{\max}$. *In particular,* $\boldsymbol{f}(\mathbf{0}, \boldsymbol{\theta}_{j^{\max}})$ *is of the form:*

$$f_j(\mathbf{0}, \boldsymbol{\theta}_{j^{\max}}) = \begin{cases} 1 & \text{if } j = j^{\max} \\ 0 & \text{otherwise,} \end{cases}$$

The proof in Appendix C.3 shows that such a $\boldsymbol{\theta}_{j^{\max}}$ exists, since $\mathbf{z}_{j^{\max}}$ is an extreme point of the polytope $\mathcal{Z}_n$. The above result shows that our estimation method recovers boundary types that consider only a single product amongst the offered products. The result also leads to the following corollary:

Corollary 3.4.1 (All extreme points $\implies$ Boundary types always optimal)**.** *If all product feature vectors* $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n$ *are extreme points of the polytope* $\mathcal{Z}_n$*, then boundary types are always optimal solutions for the* support finding step (3.5).

The above result implies that when all product features are extreme points of the polytope $\mathcal{Z}_n$, the support finding step (3.5) can be solved to optimality in each iteration, where the optimal solution corresponds to a boundary type that chooses a single product with probability 1 from amongst all offered products. Consequently, our estimation technique decomposes the population into such boundary types to explain the observed choice data. In fact, in this scenario, we can also establish the following convergence guarantee for the iterates generated by Algorithm 8:

Theorem 3.5 (Convergence in finite number of iterations)**.** *Suppose that* $\mathbf{z}_j$ *is an extreme point of the polytope* $\mathcal{Z}_n$ *for all* $j \in [n]$. *For both the* NLL *and* SQ *loss functions defined in* (3.4)*, the estimates* $\mathbf{g}^{(k)}$ *produced by Algorithm 8 converge to the optimal solution* $\mathbf{g}^*$ *in at most* n *iterations. In particular, the optimal solution* $\mathbf{g}^* = \boldsymbol{y}$ *and consequently, the CG algorithm is able to perfectly match the observed sales fractions.*

[8] This subsumes the setting of categorical features since a categorical feature is usually transformed into a set of binary features using an encoding scheme like dummy coding or one-hot coding.

Due to the complexity of the resulting optimization problems, there are few convergence guarantees for the estimation of logit models that exist in the literature. For instance, Hunter [Hun04] presents necessary and sufficient conditions for an iterative minorization-maximization (MM) algorithm to converge to the maximum likelihood estimate for a *single class* MNL model. Recently, James [Jam17] proposed an MM algorithm for estimation of mixed logit models with a multivariate normal mixing distribution, but did not provide any conditions for convergence. To the best of our knowledge, our result is one of the first to provide a convergence guarantee for general mixtures of logit models.

Some product features are binary. When some of the features are binary, the optimal solution to the support finding step always corresponds to a boundary type, having the following structure:

Theorem 3.6 (Binary feature $\implies$ Boundary types are always optimal)**.** *For each product $\ell \in [n]$, let $\mathbf{z}_\ell \in \mathbb{R}^{D_1}$ and $\mathbf{b}_\ell \in \{0,1\}^{D_2}$ represent a set of continuous and binary features respectively, where $D_1 + D_2 = D$ and $D_1, D_2 > 0$. Define the binary relation $\sim$ on $[n]$ as: $i \sim j \iff \mathbf{b}_i = \mathbf{b}_j$. Since $\sim$ is an equivalence relation on $[n]$, let $\mathcal{E}$ represent the equivalence classes, i.e. $[n] = \bigcup_{e \in \mathcal{E}} S_e$ and $S_{e_1} \cap S_{e_2} = \emptyset$ for all $e_1, e_2 \in \mathcal{E}$ such that $e_1 \neq e_2$. Then, there exists $e^* \in \mathcal{E}$ such that the optimal solution to* support finding step *(3.5) is a boundary type $\boldsymbol{f}(\boldsymbol{\omega}_0, \boldsymbol{\theta})$ where $\boldsymbol{\theta} \in \mathbb{R}^D$ satisfies*

$$\boldsymbol{\theta}^\top (\mathbf{z}_j \circ \mathbf{b}_j) > \boldsymbol{\theta}^\top (\mathbf{z}_i \circ \mathbf{b}_i) \;\; \forall\, j \in S_{e^*};\; \forall\, i \in [n] \setminus S_{e^*}$$

where $\circ$ denotes vector concatenation. In particular, $f_i(\boldsymbol{\omega}_0, \boldsymbol{\theta}) = 0$ for all $i \in [n] \setminus S_{e^*}$ so that the boundary type only considers products within subset $S_{e^*} \subset [n]$.

The proof in Appendix C.5 shows the existence of such a $\boldsymbol{\theta}$. Theorem 3.6 establishes that if products have certain binary features (in addition to continuous features), then the support finding step (3.5) always has a boundary type as the optimal solution. The consideration sets of the resulting types follow a conjunctive decision rule [Hau14], where customers screen products with a set of "must have" or "must not have" aspects—corresponding to each binary attribute—reflecting (strong) non-compensatory preferences. We can interpret the above result in the context of our sushi case study (see Section 3.6.1), where the products represent two different kinds of sushi varieties—*maki* and *non-maki*. The above result says that we recover boundary types in each iteration, each of which only consider one kind of sushi variety: either maki or non-maki. Note that the mixing distribution can contain more than one boundary type with the same consideration set, as the types will be differentiated in their choice behavior according to the parameters $(\boldsymbol{\omega}_0, \boldsymbol{\theta})$. In particular, based on the value of the parameter $\boldsymbol{\theta}$, even some products within subset S_{e^*} may *not* be considered by the boundary type. We analyze the structure of the recovered mixing distribution in more detail in the case study.

3.5 Robustness to different ground-truth mixing distributions

In this section, we use a simulation study to showcase the ability of our nonparametric estimator to obtain good approximations to various underlying mixing distributions. Our method uses only the transaction data and has no prior knowledge of the structure of the ground-truth distribution. In our study, we generate synthetic transaction data from three different ground-truth mixing distributions. We then compare the mixing distribution estimated by our method to the one estimated by a standard random parameters logit (RPL) benchmark. The benchmark is a parametric method, which makes the static assumption that the underlying mixing distribution is multivariate normal. Our results demonstrate the cost of model misspecification—the parametric RPL benchmark yields significantly poor approximations to the ground-truth mixing distributions. On the other hand, our nonparametric method is able to automatically learn from the transaction data to construct a good approximation. This specific property of our estimator makes it very appealing in practice, where one has little knowledge of the ground-truth mixing distribution.

Setup. So that we can readily compare our method to existing methods, we borrow the experimental setup from Fox et al. [FKRB11] for our simulation study. They propose a nonparametric linear regression-based estimator for recovering the mixing distribution. The key distinction from our method is that they require knowledge of the support of the mixing distribution, but our method does not. We discuss the implications of this difference towards the end of this section.

The universe consists of $n = 11$ products, one of which is the no-purchase or the outside option. Customers make choices according to a mixture of logit model with ground-truth mixing distribution Q. A customer arrives in each time period t. The firm offers all the products in the universe to the customer, but customizes the product features, offering product j with feature vector $\mathbf{z}_{jt} = (z_{jt1}, z_{jt2})$ in period t. We assume that the outside option is represented by the all zeros feature vector $(0, 0)$. The customer makes a single choice by first sampling a MNL parameter vector $\boldsymbol{\omega}^{(t)} = (\omega_1^{(t)}, \omega_2^{(t)})$ and then choosing product j with probability[9]

$$f_{jt}(\boldsymbol{\omega}^{(t)}) = \frac{\exp(\omega_1^{(t)} \cdot z_{jt1} + \omega_2^{(t)} \cdot z_{jt2})}{\sum_{\ell \in [n]} \exp(\omega_1^{(t)} \cdot z_{\ell t1} + \omega_2^{(t)} \cdot z_{\ell t2})}.$$

We consider three underlying ground-truth distributions Q:

1. Mixture of 2 bivariate Gaussians: $Q^{(2)} = 0.4 \cdot \mathcal{N}([3, 0], \boldsymbol{\Sigma}_1) + 0.6 \cdot \mathcal{N}([-1, 1], \boldsymbol{\Sigma}_2)$.
2. Mixture of 4 bivariate Gaussians:

$$Q^{(4)} = 0.2 \cdot \mathcal{N}([3, 0], \boldsymbol{\Sigma}_1) + 0.4 \cdot \mathcal{N}([0, 3], \boldsymbol{\Sigma}_1) + 0.3 \cdot \mathcal{N}([1, -1], \boldsymbol{\Sigma}_2) + 0.1 \cdot \mathcal{N}([-1, 1], \boldsymbol{\Sigma}_2)$$

[9] [FKRB11] calls this individual-level choice data.

3. Mixture of 6 bivariate Gaussians:

$$Q^{(6)} = 0.1 \cdot \mathcal{N}([3,0], \mathbf{\Sigma}_1) + 0.2 \cdot \mathcal{N}([0,3], \mathbf{\Sigma}_1) + 0.2 \cdot \mathcal{N}([1,-1], \mathbf{\Sigma}_1) \\ + 0.1 \cdot \mathcal{N}([-1,1], \mathbf{\Sigma}_2) + 0.3 \cdot \mathcal{N}([2,1], \mathbf{\Sigma}_2) + 0.1 \cdot \mathcal{N}([1,2], \mathbf{\Sigma}_2)$$

where $\mathbf{\Sigma}_1 = \begin{bmatrix} 0.2 & -0.1 \\ -0.1 & 0.4 \end{bmatrix}$ and $\mathbf{\Sigma}_2 = \begin{bmatrix} 0.3 & 0.1 \\ 0.1 & 0.3 \end{bmatrix}$ denote the variance-covariance matrices of the component Gaussian distributions.

We generated nine instances by varying the ground-truth mixing distribution Q over the set $\{Q^{(2)}, Q^{(4)}, Q^{(6)}\}$ and the number of time periods T over the set $\{2000, 5000, 10000\}$. For each combination of Q and T and time period $t \in [T]$, we generate choice data as follows: (a) we sample product features z_{jtd} according to the distribution $\mathcal{N}(0, 1.5^2)$ independently for all products $j \in [n]$, except the no purchase option, and for all features $d \in \{1, 2\}$; (b) we sample a logit parameter vector $\boldsymbol{\omega}^{(t)}$ from the ground-truth mixing distribution Q, and then (c) we generate a single choice $j \in [n]$ with probability $f_{jt}(\boldsymbol{\omega}^{(t)})$. Note that there is a single choice observation $N_t = 1$ in each time period $t \in [T]$. We replicate the above process $R = 50$ times. For each replication $r \in [R]$, we obtain mixture cumulative distribution functions (CDFs) $\hat{F}_r^{RPL}$ and $\hat{F}_r^{CG}$ by fitting the standard RPL model with a bivariate normal mixing distribution and optimizing the NLL loss using our CG algorithm, respectively. To assess the goodness of fit, we use the following two metrics proposed in [FKRB11]: root mean integrated squared error (RMISE) and mean integrated absolute error (MIAE), defined as

$$\text{RMISE} = \sqrt{\frac{1}{R}\sum_{r=1}^{R}\left[\frac{1}{V}\sum_{v=1}^{V}\left(\hat{F}_r(\boldsymbol{\beta}_v) - F_0(\boldsymbol{\beta}_v)\right)^2\right]}; \text{MIAE} = \frac{1}{V \cdot R}\sum_{r=1}^{R}\sum_{v=1}^{V}\left|\hat{F}_r(\boldsymbol{\beta}_v) - F_0(\boldsymbol{\beta}_v)\right|,$$

where $\hat{F}_r \in \{\hat{F}_r^{RPL}, \hat{F}_r^{CG}\}$, $\boldsymbol{\beta}_v$'s represent $V = 10^4$ uniformly spaced points in the rectangle $[-6, 6] \times [-6, 6]$ where the CDF is evaluated[10] and F_0 is the CDF of the ground-truth mixing distribution Q. Clearly, lower values are preferred.

Results and Discussion. Figure 3.1 and Table 3.1 summarize the results we obtained when we ran our estimator for $K_{\max} = 81$ iterations. Figure 3.1 shows a bar graph comparing our method to the RPL model on the RMISE and MIAE metrics. These metrics are compared for the three ground-truth mixing distributions, for the case with $T = 10,000$ periods. Table 3.1 shows a more complete comparison, including for the cases with $T = 2,000$ and $T = 5,000$ periods. We make the following observations:

1. Our nonparametric method is able to automatically construct a good approximation of the ground-truth mixing distribution Q from the transaction data, without any prior knowledge of the structure of Q. The benchmark RPL model, on the other hand, performs significantly worse because of model misspecification.

[10] The true mixing distribution's support lies in this region with probability close to 1.

Error metrics for the different ground-truth mixing distributions (T = 10,000 periods)

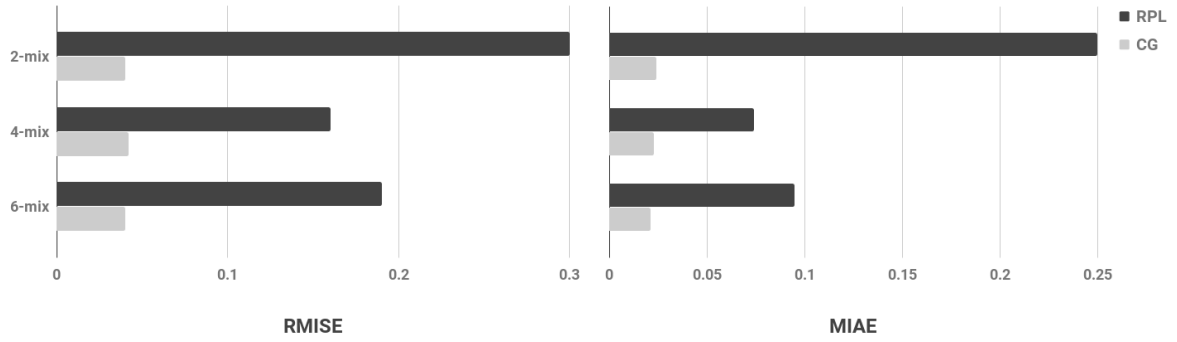

Figure 3.1: The labels 2-mix, 4-mix and 6-mix refer, respectively, to the ground-truth mixing distributions $Q^{(2)}$, $Q^{(4)}$ and $Q^{(6)}$, described in the main text. Lower values for the error metrics are preferred.

Error metrics for the different ground-truth mixing distributions as a function of the number of periods T

	RMISE						MIAE					
	2-mix		4-mix		6-mix		2-mix		4-mix		6-mix	
T	RPL	CG	RPL	CG	RPL	CG	RPL	CG	RPL	CG	RPL	CG
2,000	0.29	0.067	0.15	0.067	0.18	0.066	0.24	0.039	0.082	0.037	0.094	0.035
5,000	0.3	0.053	0.15	0.051	0.18	0.053	0.25	0.03	0.078	0.0289	0.094	0.028
10,000	0.3	0.04	0.16	0.042	0.19	0.04	0.25	0.024	0.074	0.023	0.095	0.021

Table 3.1: The metrics for the RPL benchmark are taken from Table 3 in [FKRB11]; we obtained similar numbers in our implementations. The labels 2-mix, 4-mix and 6-mix refer, respectively, to the ground-truth mixing distributions $Q^{(2)}$, $Q^{(4)}$ and $Q^{(6)}$, described in the main text. Lower values for the error metrics are preferred.

2. Table 3.1 shows that our estimator becomes better as the number of periods (and correspondingly, the samples) T increases. This improvement, which is characteristic of nonparametric estimators, shows that our method is able to extract more information as more data is made available. The RPL model, by contrast, does not exhibit any such consistent pattern.

3. Although not shown in Table 3.1, we note that the errors metrics reported by Fox et al. for their method [FKRB11, Tables 1 & 2] are comparable (or slightly worse) to those obtained under our method. Their method, however, needs the support of the mixing distribution as input. For their experiments under the simulation setup above, they use a uniform discrete grid as the support of the mixing distribution. This approach, however, does not scale to high-dimensional settings with larger D values. Our estimator does not suffer from this limitation—we show that it scales to the feature dimensions in real-world case studies, with $D = 5$ (Section 3.6.1) and $D = 11$ (Section 3.6.2).

3.6 Predictive performance of the estimator

We perform two numerical studies on real-world data to showcase the predictive accuracy of our method. The first case study uses market share data, while the second study applies our estimation technique on sales transaction data from multiple stores with varying offer-sets and product prices.

3.6.1 Case Study 1: SUSHI Preference Dataset

In this study, we compare our CG method with the expectation-maximization (EM) benchmark on the in-sample fit, and predictive and decision accuracies. Our results show that when compared to the EM benchmark, the CG method achieves 24% and 58% lower in-sample negative log-likelihood and squared loss, respectively, even when using fewer customer types. It is more than an order of magnitude (16×) faster and is 28% more accurate, on the standard root mean squared error (RMSE) metric, in predicting market shares when products are dropped, added, or replaced from the existing assortment. Finally, it can extract upto 23% more revenue from the population. These improvements over the EM benchmark are substantial, especially when it is noted that both methods are fitting the *same* model. They highlight the significance of a more accurate and scalable estimation method.

We use the popular SUSHI Preference dataset [KKA05] for our study. This dataset has been used extensively in prior work on learning customer preferences. It consists of the preferences of 5,000 customers over 100 varieties of sushi. Each customer has provided two preference orderings: (a) a complete ordering on the same subset of 10 sushi varieties and (b) a rank ordering of the top-10 of her most preferred sushi varieties from among all the 100 varieties. Each sushi variety is described by a set of features like price, oiliness in taste, frequency with which the variety is sold in the shop etc. Table 3.2 describes a subset of

Feature	Type	Range
Style	Binary	0 (maki) or 1 (otherwise)
Oiliness in taste	Continuous	[0, 4] (0: most oily)
Frequency sold in shop	Continuous	[0, 1] (1: most frequent)
Frequency of consumption	Continuous	[0, 3] (3: most frequent)
Normalized price	Continuous	[1, 5]

Table 3.2: Product features used from the SUSHI dataset

$D = 5$ features that we used in our experiments. One of the features, style, is binary valued and the rest are continuous-valued.

Setup. We processed the data to obtain aggregate market share information as follows. We assume that customers can choose from any of the 100 varieties of sushi and they choose their most preferred variety. Therefore, the market share y_j of sushi variety j is equal to the fraction of customers who ranked sushi variety j at the top. Only 93 sushi varieties had non-zero market shares, so we restrict our analysis to these varieties; therefore $n = 93$. We represent the data as the *empirical market shares* vector $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)$.

We fit a mixture of logit models to the above market share data using both our CG estimator and the EM benchmark. We initialized the CG estimator with the output of a 2-class LC-MNL model fit using the EM algorithm. For EM, we consider 5 (for NLL loss) and 10 (for SQ loss) different random initializations of the model parameters, and choose the one that attains the best loss objective. To solve the optimization problem in the support finding step, we use a heuristic algorithm that is based on our theoretical development in Section 3.4.3. The heuristic is described in detail in Appendix C.6. It obtains an approximate solution by exploiting the fact that the optimal solution to the support finding step must be a boundary type because one of the features is binary-valued (see Theorem 3.6). We run the CG algorithm for $K_{\max} = 30$ iterations so that the number of types found is at most 30. For the EM method, we vary the number K of latent classes over the set $\{2, 3, 4, 5, 10, 15, 20, 25, 30\}$ to estimate K-class LC-MNL models.

In-sample fit and structure of recovered mixing distribution. We first discuss the in-sample performance achieved by both methods. For the NLL loss, we measure the performance in terms of the KL-divergence loss, defined as $D_{KL}^{\mathsf{algo}} \stackrel{\text{def}}{=} \mathsf{NLL}^{\mathsf{algo}} - H(\boldsymbol{y})$, where $H(\boldsymbol{y}) = -\sum_{j=1}^{n} y_j \log y_j$ is the entropy of the empirical market shares vector and represents the lowest achievable in-sample NLL loss (by any method), and $\mathsf{NLL}^{\mathsf{algo}}$ denotes the NLL loss achieved by $\mathsf{algo} \in \{\text{EM}, \text{CG}\}$. Figure 3.2 plots the in-sample KL-divergence loss and squared loss as a function of the number of customer types for the EM benchmark and the number of iterations for our CG estimator. Note that the number of iterations of the CG method is an upper bound on the number of customer types it recovers. Therefore, in the comparison, the CG method is allowed to use the same number of customer types as—or even fewer than—the EM benchmark.

We make the following observations. First, the CG method consistently achieves a better

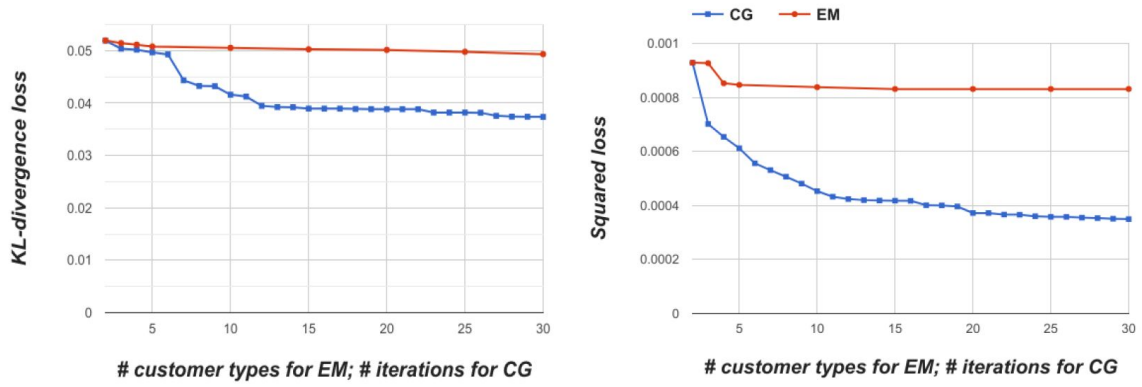

Figure 3.2: Each point on the EM curves reports the best loss achieved under different random initializations—see the main text for details. The horizontal axis represents both the number of customer types estimated by the EM benchmark, as well as the number of iterations in the CG algorithm; since the number of iterations is an upper bound on the number of types the CG estimator recovers, the plots represent a fair comparison between the methods.

in-sample fit than the EM benchmark, even when using far fewer customer types. In particular, at the end of 30 iterations, CG achieves $D_{KL}^{\text{CG}} = 0.0372$ with $K = 29$ types as opposed to $D_{KL}^{\text{EM}} = 0.049$ with $K = 30$ types—a 24% reduction. For the squared loss, CG found $K = 23$ types with an in-sample loss of 3.49×10^{-4} as opposed to 8.31×10^{-4} achieved by EM with $K = 30$ types—a 58% reduction. Second, the improvement in SQ loss is significantly higher than the improvement in NLL loss. The reason is that the M-step in the EM benchmark is non-convex when optimizing the SQ loss; consequently, it can only be solved approximately, resulting in slow convergence and worse performance for the EM benchmark. The CG algorithm, on the other hand, required very little customization[11] showing its plug-and-play nature when dealing with different loss functions.

We next analyze the structures of the customer types recovered by our method. For this analysis we focus on the NLL loss. At the end of 30 iterations, the CG method recovered 29 customer types. Except for the two customer types which were part of the initial solution, each of the remaining 27 types found by the CG method is a boundary type. These boundary types fall into two classes: those that consider only the maki (a.k.a rolled sushi) variety and those that consider only the non-maki variety. It follows from Theorem 3.6 that these are the only two possible boundary types because there is only one binary-valued feature, representing whether the sushi is maki or non-maki. Of the 93 varieties of sushi, 13 varieties are maki and the remaining 80 are non-maki. We find that 5 customer types—comprising 2.5% of the probability mass—only consider the 13 maki varieties, so if one of the maki varieties is stocked-out, they substitute to one of the remaining maki varieties. The remaining

[11] In fact, we only had to modify the objective and gradient computations.

Heatmap of choice probabilities for each sushi variety under customer types recovered by EM (left) and CG (right)

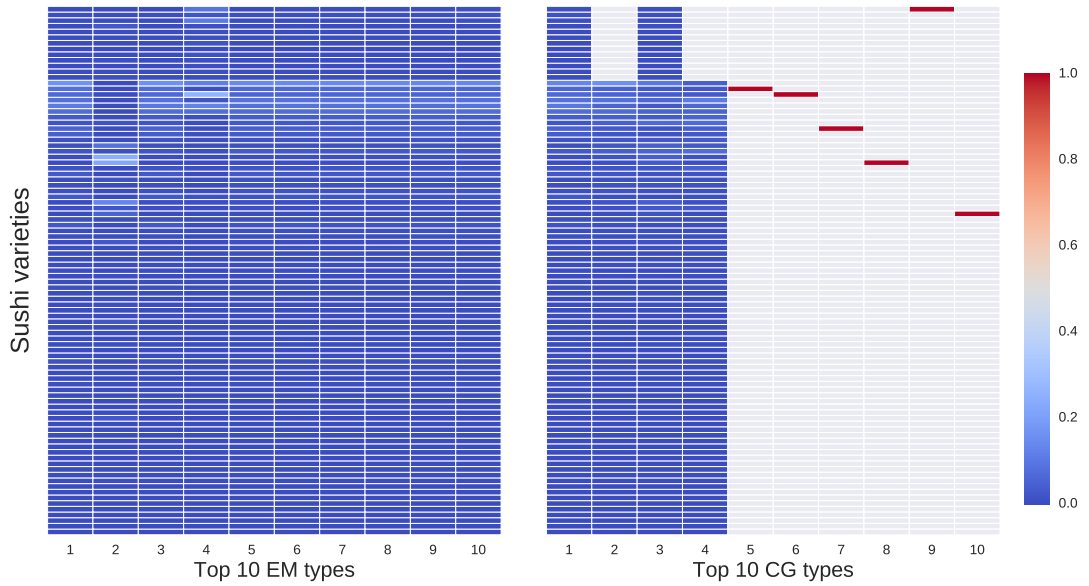

Figure 3.3: Each row corresponds to a sushi variety and each column corresponds to a customer type—for both EM and CG we choose the 10 largest types (in terms of proportions). The top 13 rows correspond to maki style sushi varieties (sorted in decreasing order of empirical market shares) and the remaining 80 rows correspond to non-maki style sushi varieties (again sorted in decreasing order of market shares). The cells depict the probability of the corresponding sushi variety being chosen by the corresponding customer type; the cells in grey correspond to sushi varieties that are not part of the consideration set, and therefore are never chosen.

22 customer types, comprising 46.1% of the probability mass, only consider the 80 non-maki varieties. The types recovered by our method exhibit strong preferences over the sushi varieties. In fact of the 10 customer types with the largest proportions, 6 types consider only a single sushi variety. By contrast, the EM algorithm recovers customer types who consider all the sushi varieties and is therefore unable to fully capture the underlying heterogeneity in the population with the same number of customer types. Figure 3.3 depicts a visual representation of this distinction, where we plot a heatmap of the choice probabilities for all the 93 sushi varieties under the recovered types for both the EM and CG estimators. As can be seen, the types recovered by EM are very similar to each other, whereas those recovered by the CG estimator appear very distinct. In particular, for the CG estimator, customer types 2 and 4 only consider non-maki style sushi varieties, types 5-8 and 10 consider only a single non-maki variety whereas type 9 only considers the maki variety with the largest market share (see the figure caption for more details).

Our theoretical characterization of the choice behavior of boundary types in Section 3.4.2 further allows managers to determine changes in sushi characteristics (such as the price) to induce maki customer types to consider non-maki varieties and vice-versa. Finally, the presence of customer types that only consider a single sushi variety is consistent with prior work where customers were observed to (consider and) purchase only a single brand of cars [LLLG95].

Predictive accuracy on new assortments. To test the predictive performance of the recovered mixture on previously unseen assortments, we consider the following tasks:

1. Predict market shares when one existing sushi variety is *dropped* from the assortment.
2. Predict market shares when one new sushi variety is *added* to the assortment.
3. Predict market shares when one existing sushi variety is *replaced* by a new variety.

The above prediction tasks are motivated by real-world situations in which products may be discontinued because of low demand and/or be unavailable due to stockouts, or new products are introduced into the market. Being able to predict how the population reacts to such changes can be very useful for a firm. We measured predictive accuracies in terms of two popular metrics, mean absolute percentage error (MAPE) and root mean-square error (RMSE), which are defined as follows: for each $\mathsf{algo} \in \{\text{EM}, \text{CG}\}$ and given any test offer-set S_{test}, we compute

$$\text{MAPE}^{\mathsf{algo}} = 100 \times \left(\frac{1}{|S_{\text{test}}|} \sum_{i \in S_{\text{test}}} \frac{\left| \hat{y}_i - \hat{y}_i^{\mathsf{algo}} \right|}{\hat{y}_i} \right); \quad \text{RMSE}^{\mathsf{algo}} = \sqrt{\frac{1}{|S_{\text{test}}|} \sum_{i \in S_{\text{test}}} \left(\hat{y}_i - \hat{y}_i^{\mathsf{algo}} \right)^2},$$

where $\hat{y}_i^{\mathsf{algo}}$ is the *predicted* market share for sushi variety $i \in S_{\text{test}}$ under the mixture of logit models[12] estimated using algo and $\hat{y}_i$ is the *true* market share computed from the test data. We report the *average error* across all possible test assortments. For the first scenario

[12]We use the mixing distribution recovered by optimizing the NLL loss.

Mixture estimation times and error in market share predictions on test assortments

Algorithm	Est. time (secs)	Drop 1		Add 1		Replace 1	
		RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
EM	914	4.6×10^{-3}	83.67	4.8×10^{-3}	90.23	4.8×10^{-3}	89.71
CG	59	3.3×10^{-3}	69.75	3.4×10^{-3}	75.06	3.5×10^{-3}	75.07
Improvement (%)	93.5	28.3	16.6	29.2	16.8	27.1	16.3

Table 3.3: "Drop 1", "Add 1" and "Replace 1" refer respectively to the cases when the test assortment is formed by dropping an existing sushi variety from the assortment, adding a new sushi variety to the assortment, and replacing an existing sushi variety with a new variety.

when one sushi variety is dropped, there are 93 test assortments resulting from dropping each sushi variety in turn. When one new variety is added, the training data consists of the market shares when 92 sushi varieties are offered to the population—we consider all $\binom{93}{2} = 93$ training assortments—and in each case, the test data consists of only a single assortment, containing all 93 varieties. We report the average error on this test assortment across each of the training assortments. Similarly, when one existing variety is replaced by a new variety, the training data consists of market shares when 92 sushi varieties are offered to the population, and for each training assortment, there are 92 test assortments—obtained by replacing each existing sushi variety in turn with a new sushi variety. We first compute the average test error for each training assortment, and finally report the mean of these average test errors across the training assortments.[13]

Table 3.3 reports the errors for each prediction task. For the EM algorithm, we choose the best performing model amongst all estimated K-class LC-MNL models. It is evident that our mixture estimation method significantly outperforms the EM benchmark across both metrics and all prediction tasks. In particular, we notice an average of 28% reduction in RMSE and 16% reduction in MAPE. Finally, we also observe from Table 3.3 that the CG method is almost 16× faster than EM-based estimation, showing that it can scale better to datasets containing large number of choice observations.

Decision accuracy. We now focus on the decision accuracies of the methods. We consider the *assortment optimization* decision, which involves determining the subset of products to offer to the population to maximize expected revenue. We find that the CG method can extract around 23% more revenue from the population, when compared to the EM benchmark.

SETUP. In order to compute the optimal assortment and ground-truth revenues, we pre-processed the data as follows: We assume that the 93 sushi varieties with non-zero market shares in the dataset comprise the entire sushi market. We focus on maximizing the revenue

[13]The improvements were similar when considering the minimum and maximum of the average test errors.

Optimal assortment sizes and revenue generated

Algorithm	# customer types recovered	Optimal assortment size	Revenue
EM	20	5	9.9×10^3
	25	5	9.9×10^3
	30	5	9.9×10^3
CG	20	17	11.9×10^3
	24	20	12.1×10^3
	28	22	12.2×10^3

Table 3.4: Note that our CG-based mixture estimation technique is able to extract around 23% more revenue compared to the EM benchmark. Here, revenue is measured in units of the normalized price feature (see Table 3.2) of each sushi variety.

from the sale of the top-49 sushi varieties by market share. The remaining 44 varieties form the outside option. Treating the outside option as one "product," we obtain a total of $n = 50$ products. Without loss of generality, we suppose that the outside option is indexed by $j = 50$. For each sushi variety $j \in [n]$, we let y_j denote its market share; for the outside option, we obtain its market share by summing the market shares of all the 44 sushi varieties it comprises. We let r_j denote the price (present as the normalized price feature in the dataset) of product j. We set the price of the outside option to 0. We suppose that the outside option is always offered. Then, our goal is to find the subset of the remaining products to maximize the expected revenue; that is, our goal is to solve

$$\max_{S \in [n-1]} \sum_{j \in S} r_j \cdot (\text{Probability that } j \text{ is chosen from } S \cup \{n\}) .$$

We fit mixtures of logit models by optimizing the NLL loss using the CG and EM methods and then solve the above optimization problem under both the models. To solve the optimization problem, we used the mixed-integer linear program (MILP) described in [MDMBVZ14]. This MILP takes as input the proportions of each mixture component, product utilities under each mixture component and the product prices, and outputs the optimal assortment. We solved the MILPs using Gurobi Optimizer version 6.5.1. The MILPs ran to optimality, so the recovered assortments were optimal for the given models.

Results and Discussion. We fit a K-class LC-MNL model using the EM method and run the CG algorithm for K iterations to estimate a mixture of logit models, where $K \in \{20, 25, 30\}$. Table 3.4 reports the optimal assortment sizes and the ground-truth revenues extracted from the population. We compute the ground-truth revenue by assuming that each of the $5,000$ customers in the dataset purchases the most preferred of the offered products, as determined from her top-10 ranking; if none of the offered products appears in the customer's top-10 ranking, then we assume that the customer chose the outside option.

We note that the EM method offers only 5 sushi varieties as part of its optimal assortment.

The reason is that the customer types recovered by the EM method are not sufficiently diverse because of which the MILP concludes that a small offering suffices to extract the most revenue from the population. In fact, the MILP ends up offering the 5 sushi varieties with the highest prices. By contrast, our method finds customer types with strong preferences who have sufficiently different tastes, so that the MILP concludes that a larger variety (around 20), consisting of both high-priced and low-priced sushi varieties, is needed in the optimal offering. The consequence is that we are able to extract around 23% more revenues from the population.

3.6.2 Case Study 2: IRI Academic Dataset

We now illustrate how our method applies to a typical operations setting in which both the offer-sets and product prices vary over time. Offer-sets vary because of stock-out events (in retail settings) and deliberate scarcity (in revenue management settings). Prices vary because of promotion activity or dynamic pricing policies. The results of our study show that when compared to the standard EM benchmark, our CG method achieves upto 8% and 7% reductions in the in-sample NLL and SQ losses, respectively, and upto 7% and 5% reductions in the out-of-sample NLL and SQ losses, respectively.

We use real-world sales transaction data from the IRI Academic Dataset [BKM08] which contains purchase transactions of consumer packaged goods (CPG) for chains of grocery and drug stores. The dataset consists of weekly sales transactions aggregated over all customers. Each transaction contains the following information: the week and store of purchase, the universal product code (UPC) of the purchased item, quantity purchased, price of the item, and whether the item was on price/display promotion. For our analysis, we consider transactions for five product categories in the first two weeks of the year 2011: shampoo, yogurt, toothbrush (toothbr), household cleaner (hhclean), and coffee. Table 3.5 describes the summary statistics of the dataset.

Setup. We consider a setup similar to that of [JR16], who used the IRI dataset to test the predictive power of their pricing method. We pre-process the raw transactions (separately for each product category), as follows. We aggregate the purchased items by vendors[14] to deal with the sparsity of the data. Then, we further aggregate the vendors into $n = 10$ "products"—one product each for the top 9 vendors with the largest market shares and a single product for all remaining vendors. This aggregation ensures that there is sufficient coverage of products in the training and test offer-sets. Next, each combination of store and week corresponds to a discrete time period t. The offer-set S_t is chosen as the union of all products purchased during the particular store-week combination. Then, for each product and offer-set pair (j, S_t), the number of sales N_{jt} is computed using the observed sales for product j in the store-week combination corresponding to S_t. The price p_{jt} of product j in offer-set S_t is set as the sales-weighted average of the prices of the different UPCs that comprise the product. The number of offer-sets obtained for each product category after this

[14]Each purchased item in the dataset is identified by its collapsed universal product code (UPC)—a 13-digit-long code with digits 4 to 8 denoting the vendor.

Product category	# Transactions	# Vendors	# Offer-sets
Shampoo	235K	168	2,464
Toothbrush	163K	122	2,462
Household cleaner	236K	217	2,470
Yogurt	544K	90	2,470
Coffee	374K	290	2,470

Table 3.5: Statistics for IRI Academic Dataset. We consider transactions in the first two weeks of the year 2011, which had a total of $1,272$ stores.

pre-processing step are listed in Table 3.5.

We assume that when offered subset S_t and prices $(p_{jt} : j \in S_t)$, each arriving customer samples the MNL parameter vector $(\boldsymbol{\mu}, \beta)$ according to some mixing distribution Q and chooses product $j \in S_t$ with probability:

$$f_{jt}(\boldsymbol{\mu}, \beta) = \frac{\exp(\mu_j - \beta \cdot p_{jt})}{\sum_{\ell \in S_t} \exp(\mu_\ell - \beta \cdot p_{\ell t})}.$$

Here, $\boldsymbol{\mu} = (\mu_1, \mu_2, \ldots, \mu_n) \in \mathbb{R}^n$ are the alternative specific coefficients, $\beta \in \mathbb{R}$ is the price coefficient. The taste vector $\boldsymbol{\omega} = (\boldsymbol{\mu}, \beta) \in \mathbb{R}^D$ with $D = n + 1 = 11$ in this context.

We fit a mixture of logit models to the processed transaction data using both the CG and EM methods. We initialize the CG algorithm with the output of a 2-class LC-MNL model fit using the EM algorithm and solve the optimization problem in the support finding step using the heuristic method described in Appendix C.6. We run the CG method for $K_{\max} = 10$ iterations, which results in a mixture with at most 10 customer types. We also fit a 10-class LC-MNL model using the EM algorithm.

Results and Discussion. Similar to [JR16], we conduct a 2-fold cross-validation. We randomly partition the offer-sets into two parts of roughly equal sizes, fit a mixture of logit model to one part (the training set), and then evaluate its predictions on the other part (the test set). We repeat this process with the train and test sets interchanged. We report performance on both the train and test datasets—all quantities referred to below are computed by taking an average across the two folds. For the NLL loss, we measure the performance using the metric

$$\Delta_{\mathsf{algo}}^{\mathsf{dataset}} = \mathsf{NLL}_{\mathsf{algo}}^{\mathsf{dataset}} - H^{\mathsf{dataset}},$$

where H^{dataset} is the sales-weighted entropy of the observed sales, defined as $H^{\mathsf{dataset}} = -\frac{1}{N}\sum_{t=1}^{T}\sum_{j \in S_t} N_{jt} \log y_{jt}$, for $\mathsf{dataset} \in \{\text{train}, \text{test}\}$, $\mathsf{algo} \in \{\text{EM}, \text{CG}\}$, and $\mathsf{NLL}_{\text{EM}}^{\mathsf{dataset}}$, $\mathsf{NLL}_{\text{CG}}^{\mathsf{dataset}}$ denote the NLL loss achieved by the EM and CG methods, respectively. In Table 3.6, we report the percentage improvement: $100 \times (\Delta_{\text{EM}}^{\mathsf{dataset}} - \Delta_{\text{CG}}^{\mathsf{dataset}})/\Delta_{\text{EM}}^{\mathsf{dataset}}$. Similarly, for the SQ loss, we report the percentage improvement as

$$100 \times (\mathsf{SQ}_{\text{EM}}^{\mathsf{dataset}} - \mathsf{SQ}_{\text{CG}}^{\mathsf{dataset}})/\mathsf{SQ}_{\text{EM}}^{\mathsf{dataset}},$$

Product category	SQ loss		NLL loss	
	Train	Test	Train	Test
Shampoo	6.4	5.1	3.4	2.3
Toothbrush	5.3	4.3	2.4	1.3
Household Cleaner	5.3	4.1	1.9	1.2
Yogurt	7.0	5.1	8.3	7.1
Coffee	4.3	2.6	3.7	2.4
Average	5.7	4.2	3.9	2.9

Table 3.6: Percentage improvements in average train/test loss over EM benchmark.

where $\mathsf{SQ}_{\mathsf{algo}}^{\mathsf{dataset}}$ denotes the SQ loss achieved by $\mathsf{algo} \in \{\text{EM}, \text{CG}\}$ on $\mathsf{dataset} \in \{\text{train}, \text{test}\}$.

Our estimator achieves better in-sample loss across all product categories and for both loss functions—an average of 5.7% reduction for SQ loss and 3.9% for the NLL loss. The in-sample improvement is largest for the yogurt category—we obtain 7.0% reduction for SQ loss and 8.3% for the NLL loss. The superior in-sample fit translates to better test performance as well, with 5.1% reduction in SQ loss for the yogurt and shampoo categories, and 7.1% reduction in NLL loss for the yogurt category.

We also analyze the structure of the recovered mixing distribution. We recovered boundary types in the mixing distribution for each of the product categories. The consideration sets for the recovered boundary types were of two kinds: (a) the type never considers a particular product (or a subset of the products) and (b) the type only considers a single product (or subset of the products). Some examples of the $\boldsymbol{\theta}$ parameters (refer to Theorem 3.3 and the subsequent discussion) of the recovered boundary types include:

1. $[0., 0., 0., -1., 0., 0., 0., 0., 0., 0., 0.]$, which means that product 4 will never be considered by this type (as long as there is another product in the offer-set).

2. $[0, 0., 0., 0.5, 0.5, 0.5, 0., 0., 0., 0.5, 0.]$, which means that products in the set $\{4, 5, 6, 10\}$ are strictly preferred to all other products, and the type will only choose amongst them (provided at least one of them is in the offer-set).

3. $[0., 0., -0.707, 0., -0.707, 0., 0., 0., 0., 0., 0.]$ which means that product 3 and/or 5 will never be considered by the type (as long as there is some other product in the offer-set).

3.7 Extension: accounting for endogeneity in product features

In many applications of discrete choice modeling, a product feature may be correlated with features not included in the model. The omitted features tend to be those that are unobserved.

If such correlations are ignored during estimation, then the coefficient estimated for the included feature could be biased. This phenomenon is referred to broadly as *endogeneity*. The classical example is that product prices are often correlated with unobservables, such as product quality, and ignoring such unobservables may lead one to conclude that higher prices lead to higher demands, when in fact, the higher demand was caused by higher quality. [PT10] offers other examples of endogeneity.

Several techniques have been proposed in existing literature to deal with the issue of endogeneity in discrete choice models. In this section, we show how one such technique can be incorporated into our method. We use the *control function* method proposed by Petrin and Train [PT10], which generalizes the demand shocks approach proposed in [BLP95]. We illustrate its use in our method using the following modification of the simulation setup from Section 3.5:

Utility model. We follow the setup of Section 3.5. We fix a choice of the ground-truth mixing distribution Q and number of time periods T. We then generate the choice data as follows. In each period $t \in [T]$, a customer arrives and is offered all the $n = 11$ products, including the no-purchase option. Instead of sampling a two-dimensional parameter vector as before, the customer now samples a three-dimensional parameter vector $(\omega_1^{(t)}, \omega_2^{(t)}, \omega_3^{(t)})$ according to Q and assigns the following utility to product j:

$$U_{jt} = \omega_1^{(t)} \cdot x_{jt} + \omega_2^{(t)} \cdot z_{jt} + \omega_3^{(t)} \cdot \mu_{jt} + \varepsilon_{jt},$$

where $(x_{jt}, z_{jt}, \mu_{jt})$ is the feature vector of product j in period t and $(\varepsilon_{jt} \colon j \in [n])$ are independent and identically distributed standard Gumbel random variables. Customers choose the product with the highest utility, resulting in the standard MNL choice probability. The key difference in the utility model from the setup above is that while x_{jt} and z_{jt} are observed, μ_{jt} is unobserved and is correlated with x_{jt}. As is standard in the literature, we assume that the endogenous feature is impacted by a set of instruments $\mathbf{w}$ and the exogenous feature:

$$x_{jt} = \gamma_1 \cdot \mathrm{w}_{jt,1} + \gamma_2 \cdot \mathrm{w}_{jt,2} + \gamma_3 \cdot z_{jt} + \mu_{jt}.$$

Control function correction. To deal with endogeneity, the control function (CF) approach obtains a proxy for the term μ_{jt} by regressing the endogenous feature x_{jt} on the instruments $(\mathrm{w}_{jt,1}, \mathrm{w}_{jt,2})$ and the exogenous feature z_{jt} and then plugs in the residual $\hat{\mu}_{jt} = x_{jt} - \hat{\boldsymbol{\gamma}}^\top (\mathrm{w}_{jt,1}, \mathrm{w}_{jt,2}, z_{jt})$, where $\hat{\boldsymbol{\gamma}}$ represents the estimated regression parameters. In other words, the method estimates the coefficients using the following utility model:

$$\hat{U}_{jt} = \omega_1^{(t)} \cdot x_{jt} + \omega_2^{(t)} \cdot z_{jt} + \omega_3^{(t)} \cdot \hat{\mu}_{jt} + \varepsilon_{jt}.$$

Once we plug in the residual, the estimators are run as before. They now estimate a mixing distribution over $D = 3$ parameters, where the additional random parameter is for the unobservable μ_{jt}.

Setup. For our experiments, we sample $(\omega_1^{(t)}, \omega_2^{(t)})$ according to the distribution $Q^{(2)}$, which is a mixture of two bivariate Gaussians, as defined in Section 3.5. We sample $\omega_3^{(t)}$ according to $\mathcal{N}(-1, 0.3^2)$, independently of $\omega_1^{(t)}$ and $\omega_2^{(t)}$. For each time period t and product

Recovery metrics under endogenous product features

	RMISE		MIAE	
Estimator	Without CF	With CF	Without CF	With CF
RPL	0.121	0.095	0.057	0.046
CG	0.074	0.059	0.039	0.038

Table 3.7: "Without CF" refers to the case when endogeneity is ignored and "With CF" refers to the case when control function (CF) correction is applied. All differences are statistically significant at 1% significance level according to a paired samples t-test.

j, we sample the exogenous feature z_{jt} according to $\mathcal{N}(0, 1.5^2)$, the instruments $\mathrm{w}_{jt1}, \mathrm{w}_{jt2}$ according to $\mathcal{N}(0,1)$, and the unobservable μ_{jt} according to $\mathcal{N}(0,1)$, all independently of each other. We choose $\boldsymbol{\gamma} = (0.54, 0.54, 0.54)$ to ensure that the marginal distribution of x_{jt} matches the marginal distribution of the features for the case without endogeneity in Section 3.5. Then, we generate choices for $T = 15,000$ periods.

Results. Table 3.7 compares our CG method to the standard RPL model with a diagonal variance-covariance matrix on the same RMISE and MIAE metrics, both when endogeneity is ignored and when endogeneity is corrected using the CF approach. We compute the error metrics only for the distribution of (ω_1, ω_2), and not ω_3. We make the following observations:

1. Ignoring endogeneity can worsen the recovery of the underlying mixing distribution, as is evident in the noticeably larger RMISE value for the benchmark RPL model.

2. Misspecification in the mixing distribution can impact recovery more adversely than ignoring endogeneity. Our method without the CF correction has lower error metrics than the benchmark *with* the CF correction. This shows that having the freedom of choosing the mixing distribution can help mitigate the effects of endogeneity bias.

3. Our estimator is compatible with the CF approach, allowing one to correct for endogeneity and obtain a better approximation to the underlying mixing distribution.

Conclusions and Future Directions

This dissertation revisited the classical problems of customer segmentation and demand learning but in the presence of sparse, diverse and large-scale data, and proposed methodologies to deal with the challenges posed by such data. We first presented a novel method to segment (or cluster) a large population of customers based on their preferences over a large collection of items, when the preference observations come from diverse data sources such as purchases, ratings, clicks, etc. and are highly sparse, i.e. each customer may provide very few observations. Then, we focused on the problem of segmentation in the presence of unreliable data and proposed algorithms to segment workers in crowdsourced labeling tasks based on their (unknown) reliability, using only the labels submitted by the workers. Finally, we proposed a nonparametric estimator for the mixture of logit model—commonly used to model the customer demand—and demonstrated its favorable properties as compared to existing parametric estimators.

There are several opportunities and directions for future work. We outline some of them below:

- For the problem of segmenting customers, we focused primarily on categorical labels (clicks, purchases, like/dislikes, etc.) since most of the observations collected about customers from firms are categorical. However, our methodology can, in principle, be applied to numerical, i.e. real-valued, observations (such as the number of clicks/purchases, time spent browsing, dollars spent etc.) and it will be interesting to explore how our algorithm performs when there is a mix of categorical and numerical observations. From the analytical perspective, it will be interesting to determine other generative models (especially from the exponential family) for customer labels under which our algorithm can recover the true segments. For instance, we could consider mixtures of binary logit models where each item j is represented using a vector $\boldsymbol{y}_j$ in some feature space $\mathcal{Y}$. Imposing suitable constraints on the space $\mathcal{Y}$ as well as defining appropriate missing data mechanisms for the customer labels will be important in this regard. More broadly, the idea of separating customers based on their deviation from the population's preferences can be applied in other contexts, such as textual reviews and restaurant photos in Yelp, to obtain interesting domain-specific notions of mainstream and esoteric segments. Finally, it would be useful to test the effectiveness of our segmentation

method in terms of standard marketing performance measures such as customer lifetime value, profitability, loyalty, etc.

- To the best of our knowledge, our work was the first to consider general worker strategies in crowdsourced labeling tasks, and has resulted in follow-up papers that explicitly model adversarial behavior (such as correlated errors and sybil attacks) in crowdsourced services [WWW+16, YLLZ17, KA18] as well as test the robustness of existing label aggregation algorithms to adversarial attacks [MLS+18]. Our theoretical analysis assumed that the tasks are homogeneous, and analyzing our reputation algorithms under models that account for heterogeneous tasks, such as variation in task difficulty (for example [WRW+09] and [WBBP10]) is a natural next step. Both of our penalty-based algorithms assumed that the task labels are binary; analyzing natural extensions of our algorithms to multi-class settings is an interesting direction. Because our algorithm allows the identification of adversarial workers, it could be combined with adaptive techniques to recruit more workers [RPGMP12] and identify the right workers [LZF14]. Finally, applying our reputation algorithms to ensemble learning approaches, in which outputs from multiple learning algorithms are combined, could be a promising future direction; indeed there has already been some work in this space [WY14].

- We focused on estimating the mixture of logit models because of its widespread use, but our approach is directly applicable for general mixture models including other choice model families—with the caveat that the subproblems in the conditional gradient algorithm can be solved reasonably efficiently. Applying our nonparametric estimator in the context of other popular choice models like the probit and Mallows models, and deriving insights for the recovered customer types, is a promising direction for future work. In fact, our framework can also be used to directly learn distributions over preference orderings (or rankings) over the products, resulting in a fully nonparametric approach. In that case, the subproblem in each iteration corresponds to finding a single ranking which has connections to the learning to rank [Liu09] and rank aggregation [DKNS01] literatures. In fact, [JR17] also use the Frank-Wolfe algorithm to estimate a distribution over preference orderings, but their setup did not consider any product features. Extending their approach to account for product features is also a promising future direction. Finally, our estimation method currently cannot account for fixed parameters (across customer types) in the utility specification. Incorporating fixed effects into the estimation framework could be an important next step for promoting more widespread adoption of our estimator.

Appendix A

Chapter 1 Proofs and Details of Numerical Experiments

A.1 Proofs of Theoretical Results

We begin by proving some general statements about random variables, that will be used in the proofs later.

Lemma A.1. *Let $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_r, \boldsymbol{Y}$ be a collection of non-negative random variables. Let $a_1, a_2, \ldots, a_r, b$ be positive constants. Then, given any $0 < \varepsilon < 1$ and any $1 \leq i \leq r$, we have that*

$$\text{(i)}\ \mathbb{P}\left[\left|\frac{\boldsymbol{X}_i}{\boldsymbol{Y}} - \frac{a_i}{b}\right| > \varepsilon\frac{a_i}{b}\right] \leq \mathbb{P}\Big[|\boldsymbol{X}_i - a_i| > \varepsilon' a_i\Big] + \mathbb{P}\Big[|\boldsymbol{Y} - b| > \varepsilon' b\Big]$$

$$\text{(ii)}\ \mathbb{P}\Big[|\boldsymbol{X}_i\boldsymbol{Y} - a_i b| > \varepsilon a_i b\Big] \leq \mathbb{P}\Big[|\boldsymbol{X}_i - a_i| > \varepsilon' a_i\Big] + \mathbb{P}\Big[|\boldsymbol{Y} - b| > \varepsilon' b\Big]$$

$$\text{(iii)}\ \mathbb{P}\left[\left|\sum_{i=1}^r \boldsymbol{X}_i - \sum_{i=1}^r a_i\right| > \varepsilon \cdot \left(\sum_{i=1}^r a_i\right)\right] \leq \sum_{i=1}^r \mathbb{P}\Big[|\boldsymbol{X}_i - a_i| > \varepsilon a_i\Big]$$

$$\text{(iv)}\ \mathbb{P}\left[\sum_{i=1}^r |\boldsymbol{X}_i - a_i| > \varepsilon \cdot \left(\sum_{i=1}^r a_i\right)\right] \leq \sum_{i=1}^r \mathbb{P}\Big[|\boldsymbol{X}_i - a_i| > \varepsilon a_i\Big]$$

where $\varepsilon' = \varepsilon/3$.

Proof. We prove parts (i) and (ii) assuming (w.l.o.g) that $i = 1$.

Part (i). Let $\boldsymbol{Z}_1 = \frac{\boldsymbol{X}_1}{\boldsymbol{Y}}$. We prove the result by contradiction. Suppose $\boldsymbol{Z}_1 > (1+\varepsilon)\frac{a_1}{b}$. Then, $\boldsymbol{X}_1 > (1+\varepsilon')a_1$ or $\boldsymbol{Y} < (1-\varepsilon')b$. If not, we have the following:

$$\begin{aligned}\boldsymbol{X}_1 \leq (1+\varepsilon')a_1,\ \ \boldsymbol{Y} \geq (1-\varepsilon')b \implies \frac{\boldsymbol{X}_1}{\boldsymbol{Y}} &\leq \frac{(1+\varepsilon')a_1}{(1-\varepsilon')b} \\ \implies \boldsymbol{Z}_1 &\leq (1+\varepsilon)\frac{a_1}{b}\end{aligned}$$

where the last implication follows from the fact that $\frac{1+\varepsilon'}{1-\varepsilon'} \leq 1+\varepsilon$ when $\varepsilon' = \varepsilon/3$ and $0 < \varepsilon < 1$. This is a contradiction. Therefore we have that,

$$\mathbb{P}\Big[\boldsymbol{Z}_1 > (1+\varepsilon)\frac{a_1}{b}\Big] \leq \mathbb{P}\Big[\boldsymbol{X}_1 > (1+\varepsilon')a_1 \bigcup \boldsymbol{Y} < (1-\varepsilon')b\Big] \leq \mathbb{P}\Big[\boldsymbol{X}_1 > (1+\varepsilon')a_1\Big] + \mathbb{P}\Big[\boldsymbol{Y} < (1-\varepsilon')b\Big]$$

where the last inequality follows from the union bound. An analogous argument establishes that

$$\mathbb{P}\Big[\boldsymbol{Z}_1 < (1-\varepsilon)\frac{a_1}{b}\Big] \leq \Big(\mathbb{P}\Big[\boldsymbol{X}_1 < (1-\varepsilon')a_1\Big] + \mathbb{P}\Big[\boldsymbol{Y} > (1+\varepsilon')b\Big]\Big)$$

which uses the fact that $\frac{1-\varepsilon'}{1+\varepsilon'} \geq 1-\varepsilon$ when $\varepsilon' = \varepsilon/3$ and $0 < \varepsilon < 1$. Combining the above two arguments, we get

$$\mathbb{P}\Big[\Big|\boldsymbol{Z}_1 - \frac{a_1}{b}\Big| > \varepsilon\frac{a_1}{b}\Big] \leq \Big(\mathbb{P}\Big[|\boldsymbol{X}_1 - a_1| > \varepsilon' a_1\Big] + \mathbb{P}\Big[|\boldsymbol{Y} - b| > \varepsilon' b\Big]\Big)$$

Part (ii). Let $\boldsymbol{W}_1 = \boldsymbol{X}_1\boldsymbol{Y}$. Suppose $\boldsymbol{W}_1 > (1+\varepsilon)a_1 b$. Then, $\boldsymbol{X}_1 > (1+\varepsilon')a_1$ or $\boldsymbol{Y} > (1+\varepsilon')b$. If not, we have the following:

$$\begin{aligned} \boldsymbol{X}_1 \leq (1+\varepsilon')a_1, \;\; \boldsymbol{Y} \leq (1+\varepsilon')b &\implies \boldsymbol{X}_1\boldsymbol{Y} \leq (1+\varepsilon')^2 a_1 b \\ &\implies \boldsymbol{W}_1 \leq (1+\varepsilon)a_1 b \end{aligned}$$

where the last implication follows because $(1+\varepsilon')^2 \leq 1+\varepsilon$ for $\varepsilon' = \varepsilon/3$ and $0 < \varepsilon < 1$. This is a contradiction. Therefore,

$$\mathbb{P}\Big[\boldsymbol{W}_1 > (1+\varepsilon)a_1 b\Big] \leq \mathbb{P}\Big[\boldsymbol{X}_1 > (1+\varepsilon')a_1 \bigcup \boldsymbol{Y} > (1+\varepsilon')b\Big] \leq \mathbb{P}\Big[\boldsymbol{X}_1 > (1+\varepsilon')a_1\Big] + \mathbb{P}\Big[\boldsymbol{Y} > (1+\varepsilon')b\Big]$$

Combining with the symmetric case gives:

$$\mathbb{P}\Big[|\boldsymbol{W}_1 - a_1 b| > \varepsilon a_1 b\Big] \leq \Big(\mathbb{P}\Big[|\boldsymbol{X}_1 - a_1| > \varepsilon' a_1\Big] + \mathbb{P}\Big[|\boldsymbol{Y} - b| > \varepsilon' b\Big]\Big)$$

Part (iii). Define $\boldsymbol{Z} \stackrel{\text{def}}{=} \sum_{i=1}^r \boldsymbol{X}_i$ and $A \stackrel{\text{def}}{=} \sum_{i=1}^r a_i$. Suppose that $\boldsymbol{Z} > (1+\varepsilon)A$. Then it follows that for some $1 \leq i \leq r$, $\boldsymbol{X}_i > (1+\varepsilon)a_i$. If not, we have:

$$\begin{aligned} \boldsymbol{X}_1 \leq (1+\varepsilon)a_1, \ldots \boldsymbol{X}_r \leq (1+\varepsilon)a_r &\implies \sum_{i=1}^r \boldsymbol{X}_i \leq \sum_{i=1}^r (1+\varepsilon)a_i \\ &\implies \boldsymbol{Z} \leq (1+\varepsilon)A \end{aligned}$$

which is a contradiction. Combining with the symmetric case and applying the union bound, the claim follows.

Part (iv). Let $\boldsymbol{Z} \stackrel{\text{def}}{=} \sum_{i=1}^{r} |\boldsymbol{X}_i - a_i|$ and suppose $\boldsymbol{Z} > \varepsilon \cdot A$, where recall from part (iii) above that $A = \sum_{i=1}^{r} a_i$. Then it follows that for some $1 \leq i \leq r$, $|\boldsymbol{X}_i - a_i| > \varepsilon a_i$. If not, we have:

$$|\boldsymbol{X}_1 - a_1| \leq \varepsilon a_1, \ldots |\boldsymbol{X}_r - a_r| \leq \varepsilon a_r \implies \sum_{i=1}^{r} |\boldsymbol{X}_i - a_i| \leq \sum_{i=1}^{r} \varepsilon a_i$$
$$\implies \boldsymbol{Z} \leq \varepsilon \cdot \left(\sum_{i=1}^{r} a_i \right) = \varepsilon \cdot A$$

which is a contradiction. The claim then follows from the union bound. □

Lemma A.2. *Let $\boldsymbol{X}, \boldsymbol{Y}$ be two non-negative random variables such that $0 \leq \boldsymbol{X}, \boldsymbol{Y} \leq 1$ and $\boldsymbol{Y} = 0 \implies \boldsymbol{X} = 0$. Suppose that $1 > \mathbb{E}[\boldsymbol{X}], \mathbb{E}[\boldsymbol{Y}] > 0$. Define the random variables $\boldsymbol{Z}_1 = \boldsymbol{X} \cdot (-\log \boldsymbol{Y})$ and $\boldsymbol{Z}_2 = \boldsymbol{X} \cdot (-\log \boldsymbol{X})$, and the constants $A_1 = \mathbb{E}[\boldsymbol{X}] \cdot (-\log \mathbb{E}[\boldsymbol{Y}])$ and $A_2 = \mathbb{E}[\boldsymbol{X}] \cdot (-\log \mathbb{E}[\boldsymbol{X}])$. Then, given any $0 < \varepsilon < 1$, we have:*

$$\mathbb{P}\left[|\boldsymbol{Z}_1 - A_1| > \varepsilon A_1 \right] \leq \mathbb{P}\left[|\boldsymbol{X} - \mathbb{E}[\boldsymbol{X}]| > \frac{\varepsilon}{3} \mathbb{E}[\boldsymbol{X}] \right] + \mathbb{P}\left[|-\log \boldsymbol{Y} - (-\log \mathbb{E}[\boldsymbol{Y}])| > \frac{\varepsilon}{3} \cdot (-\log \mathbb{E}[\boldsymbol{Y}]) \right]$$

$$\mathbb{P}\left[|\boldsymbol{Z}_2 - A_2| > \varepsilon A_2 \right] \leq \mathbb{P}\left[|\boldsymbol{X} - \mathbb{E}[\boldsymbol{X}]| > \frac{\varepsilon}{3} \mathbb{E}[\boldsymbol{X}] \right] + \mathbb{P}\left[|-\log \boldsymbol{X} - (-\log \mathbb{E}[\boldsymbol{X}])| > \frac{\varepsilon}{3} \cdot (-\log \mathbb{E}[\boldsymbol{X}]) \right]$$

Proof. Note that since $\boldsymbol{X}, \boldsymbol{Y} \in [0, 1]$, it follows that $-\log \boldsymbol{X}, -\log \boldsymbol{Y}$ are non-negative. Also, since $\boldsymbol{Y} = 0 \implies \boldsymbol{X} = 0$, the random variables $\boldsymbol{Z}_1, \boldsymbol{Z}_2$ are both well-defined (with the convention that $x \cdot \log x = 0$ when $x = 0$). Further, since we also have $0 < \mathbb{E}[\boldsymbol{X}], \mathbb{E}[\boldsymbol{Y}] < 1$, so that $-\log \mathbb{E}[\boldsymbol{X}] > 0$ and $-\log \mathbb{E}[\boldsymbol{Y}] > 0$. The claims then follow from a straightforward application of part (ii) of Lemma A.1. □

Lemma A.3. *Let $0 \leq \boldsymbol{X} \leq 1$ be a non-negative random variable with $1 > \mathbb{E}[\boldsymbol{X}] > 0$. Then given any $0 < \varepsilon < 1$, for all values of the random variable $\boldsymbol{X}$ in the interval $I := \left((1 - \varepsilon')\mathbb{E}[\boldsymbol{X}], (1 + \varepsilon')\mathbb{E}[\boldsymbol{X}] \right)$, where $\varepsilon' = \frac{-\varepsilon \log \mathbb{E}[\boldsymbol{X}]}{1 - \varepsilon \log \mathbb{E}[\boldsymbol{X}]}$, we have*

$$|-\log(\boldsymbol{X}) - (-\log \mathbb{E}[\boldsymbol{X}])| \leq \varepsilon \cdot (-\log \mathbb{E}[\boldsymbol{X}])$$

Proof. Since $0 < \mathbb{E}[\boldsymbol{X}] < 1$, it means that $-\log \mathbb{E}[\boldsymbol{X}] > 0$ and consequently, $0 < \varepsilon' < 1$. In addition, it can be seen that $(1 + \varepsilon')\mathbb{E}[\boldsymbol{X}] \leq 1$ for any $0 < \varepsilon < 1$ and any $0 < \mathbb{E}[\boldsymbol{X}] < 1$, so that $I \subset [0, 1]$. Consider the function $g(x) = -\log x$ and note that it is continuous and differentiable on the interval I. The Mean Value theorem says that given a differentiable function $g(\cdot)$ in the interval (a, b), there exists $c \in (a, b)$ such that

$$\frac{g(b) - g(a)}{b - a} = g'(c) = \frac{g(a) - g(b)}{a - b}$$

where $g'(\cdot)$ is the derivative of $g(\cdot)$. Using the mean value theorem for $g(x) = -\log x$ in the interval I, it follows that for all values of random variable $\boldsymbol{X} \in I$ there exists some $\boldsymbol{Z}$ between $\mathbb{E}[\boldsymbol{X}]$ and $\boldsymbol{X}$ such that

$$\frac{-\log \boldsymbol{X} - (-\log \mathbb{E}[\boldsymbol{X}])}{\boldsymbol{X} - \mathbb{E}[\boldsymbol{X}]} = \frac{-1}{\boldsymbol{Z}}$$

Now since $\boldsymbol{Z} \in I$, it follows that $\frac{1}{\boldsymbol{Z}} \leq \frac{1}{(1-\varepsilon')\mathbb{E}[\boldsymbol{X}]}$. Also, since $\boldsymbol{X} \in I$, we have $|\boldsymbol{X} - \mathbb{E}[\boldsymbol{X}]| \leq \varepsilon'\mathbb{E}[\boldsymbol{X}]$. Then it follows:

$$\begin{aligned}
|-\log \boldsymbol{X} - (-\log \mathbb{E}[\boldsymbol{X}])| &= \left|\frac{-(\boldsymbol{X} - \mathbb{E}[\boldsymbol{X}])}{\boldsymbol{Z}}\right| \\
&= \frac{|\boldsymbol{X} - \mathbb{E}[\boldsymbol{X}]|}{\boldsymbol{Z}} \\
&\leq \frac{|\boldsymbol{X} - \mathbb{E}[\boldsymbol{X}]|}{(1-\varepsilon')\mathbb{E}[\boldsymbol{X}]} \leq \frac{\varepsilon'}{1-\varepsilon'} = \varepsilon \cdot (-\log \mathbb{E}[\boldsymbol{X}])
\end{aligned}$$

□

A.1.1 Proofs of Section 1.3.1

First, we introduce some additional notation. Recall from the main text that $\mathbf{1}[A]$ denotes the indicator variable taking value 1 if an event A is true and 0 otherwise. Let $\boldsymbol{X}_i^+$ (resp. $\boldsymbol{X}_i^-$) denote the number of items rated as $+1$ (resp. -1) by customer i. In other words, $\boldsymbol{X}_i^+ = \sum_{j \in N(i)} \mathbf{1}[\boldsymbol{X}_{ij} = +1]$ and $\boldsymbol{X}_i^- = \sum_{j \in N(i)} \mathbf{1}[\boldsymbol{X}_{ij} = -1]$, where recall that $N(i)$ denotes the set of items rated by customer i. Here, $\boldsymbol{X}_{ij}$ represents the rating provided by customer i for item j, note that it is a random variable under the LC-IND model. Next, let $\boldsymbol{F}_0^+ = \frac{\sum_{i'=1}^m \boldsymbol{X}_{i'}^+}{m \cdot \ell}$, so $\boldsymbol{F}_0^+$ is the fraction of likes (+1s) received from the customer population. Finally, let $\mathrm{Bin}(r, p)$ denote the Binomial distribution with parameters r and p.

We begin by establishing a lemma that will be used in the proof later.

Lemma A.4. *Consider the random variable $\boldsymbol{F}_0^+ = \frac{\sum_{i'=1}^m \boldsymbol{X}_{i'}^+}{m \cdot \ell}$. Given any $t > 0$, the following facts are true:*

$$\text{(i) } \mathbb{E}[\boldsymbol{F}_0^+] = \alpha_{\text{pool}} \qquad \text{(ii) } \mathbb{P}\Big[\,\big|\boldsymbol{F}_0^+ - \alpha_{\text{pool}}\big| \geq t\,\Big] \leq 2\exp\big(-2m\ell t^2\big)$$

Proof. We begin with the expectation:

$$\mathbb{E}[\boldsymbol{F}_0^+] = \frac{\sum_{i'=1}^m \mathbb{E}[\boldsymbol{X}_{i'}^+]}{m \cdot \ell} = \frac{\sum_{i'=1}^m \ell \alpha_{z_i'}}{m \cdot \ell} = \frac{\sum_{k'=1}^K q_{k'} m \cdot (\ell \alpha_{k'})}{m \cdot \ell} = \sum_{k'=1}^K q_{k'} \alpha_{k'} = \alpha_{\text{pool}}$$

where the third equality follows from the fact that proportion $q_{k'}$ of the customer population belongs to segment k'. For part (ii), observe that $\boldsymbol{F}_0^+$ can be equivalently written as:

$$\boldsymbol{F}_0^+ = \frac{\sum_{i'=1}^{m} \sum_{j \in N(i')} \mathbf{1}[\boldsymbol{X}_{i'j} = +1]}{m \cdot \ell}.$$

In other words, $\boldsymbol{F}_0^+$ is an average of $m \cdot \ell$ random variables, which are independent under the LC-IND model (since each customer rates items independently). Then using Hoeffding's inequality we can show, for any $t > 0$:

$$\mathbb{P}\Big[\big|\boldsymbol{F}_0^+ - \alpha_{\text{pool}}\big| \geq t\Big] \leq 2\exp\big(-2m\ell t^2\big).$$

□

A.1.2 Proof of Lemma 1.1

To calculate the customer embedding scores, we first need to compute the pooled estimate. Since the underlying LC-IND model is parameterized by a single parameter that specifies the probability of liking any item, the pooled estimate is given by $\frac{\sum_{i'=1}^{m} \boldsymbol{X}_{i'}^+}{m \cdot \ell} = \boldsymbol{F}_0^+$ based on our definition earlier. Also, let us denote the fraction of likes given by customer i as $\boldsymbol{F}_i^+ \stackrel{\text{def}}{=} \frac{\boldsymbol{X}_i^+}{\ell}$. Then, the unidimensional embedding $\boldsymbol{V}_i$ for customer i is given by:

$$\begin{aligned}
\boldsymbol{V}_i &= \frac{-\sum_{j \in N(i)} \big(\mathbf{1}[\boldsymbol{X}_{ij} = +1] \log \boldsymbol{F}_0^+ + \mathbf{1}[\boldsymbol{X}_{ij} = -1] \log(1 - \boldsymbol{F}_0^+)\big)}{-\sum_{j \in N(i)} \big(\boldsymbol{F}_0^+ \log \boldsymbol{F}_0^+ + (1 - \boldsymbol{F}_0^+) \log(1 - \boldsymbol{F}_0^+)\big)} \\
&= \frac{-\big(\sum_{j \in N(i)} \mathbf{1}[\boldsymbol{X}_{ij} = +1]\big) \log \boldsymbol{F}_0^+ - \big(\sum_{j \in N(i)} \mathbf{1}[\boldsymbol{X}_{ij} = -1]\big) \log(1 - \boldsymbol{F}_0^+)}{-\big(\boldsymbol{F}_0^+ \log \boldsymbol{F}_0^+ + (1 - \boldsymbol{F}_0^+) \log(1 - \boldsymbol{F}_0^+)\big) \cdot \ell} \\
&= \frac{-\big(\frac{\boldsymbol{X}_i^+}{\ell}\big) \log \boldsymbol{F}_0^+ - \big(1 - \frac{\boldsymbol{X}_i^+}{\ell}\big) \log(1 - \boldsymbol{F}_0^+)}{-\big(\boldsymbol{F}_0^+ \log \boldsymbol{F}_0^+ + (1 - \boldsymbol{F}_0^+) \log(1 - \boldsymbol{F}_0^+)\big)} \\
&= \frac{-\boldsymbol{F}_i^+ \log \boldsymbol{F}_0^+ - \big(1 - \boldsymbol{F}_i^+\big) \log(1 - \boldsymbol{F}_0^+)}{-\big(\boldsymbol{F}_0^+ \log \boldsymbol{F}_0^+ + (1 - \boldsymbol{F}_0^+) \log(1 - \boldsymbol{F}_0^+)\big)}
\end{aligned}$$

Note that when $\boldsymbol{F}_0^+ \in \{0, 1\}$, we define $\boldsymbol{V}_i = 0$ which is the limiting value as $\boldsymbol{F}_0^+ \to 0$ or $\boldsymbol{F}_0^+ \to 1$.

Concentration of $\boldsymbol{F}_i^+$. From the generative model, it follows that the random variable representing the number of likes given by customer i is a binomial random variable, i.e. $\boldsymbol{X}_i^+ \sim \text{Bin}(\ell, \alpha_{z_i})$. Then, using Hoeffding's inequality we can show that for any $t > 0$:

$$\begin{aligned}
\mathbb{P}\Big[\big|\boldsymbol{F}_i^+ - \alpha_{z_i}\big| \geq t\Big] &\leq 2\exp\big(-2\ell t^2\big) \\
\mathbb{P}\Big[\big|\big(1 - \boldsymbol{F}_i^+\big) - (1 - \alpha_{z_i})\big| \geq t\Big] &\leq 2\exp\big(-2\ell t^2\big) \qquad \text{(A.1)}
\end{aligned}$$

Concentration of $-\log \boldsymbol{F}_0^+$ and $-\log(1-\boldsymbol{F}_0^+)$. Lemma A.4 says that $\mathbb{E}[\boldsymbol{F}_0^+] = \alpha_{\text{pool}} \geq \alpha_{\min} > 0$ and observe that $0 \leq \boldsymbol{F}_0^+ \leq 1$. So we can apply Lemma A.3 to the random variable $\boldsymbol{F}_0^+$, which says that given any $0 < \varepsilon < 1$, for all values of random variable $\boldsymbol{F}_0^+$ in the interval $\Big(\alpha_{\text{pool}} \cdot (1-\varepsilon'), \alpha_{\text{pool}} \cdot (1+\varepsilon')\Big)$ with $\varepsilon' = \frac{-\varepsilon \log \alpha_{\text{pool}}}{1-\varepsilon \log \alpha_{\text{pool}}}$, we have:

$$\big|-\log \boldsymbol{F}_0^+ - (-\log \alpha_{\text{pool}})\big| \leq \varepsilon \cdot (-\log \alpha_{\text{pool}}) \tag{A.2}$$

Now, for any $0 < \varepsilon < 1$, define $t(\varepsilon) \stackrel{\text{def}}{=} \varepsilon \cdot \left(\frac{-\bar{\alpha}_{\text{pool}} \cdot \log\left(1-\bar{\alpha}_{\text{pool}}\right)}{1-\log\left(1-\bar{\alpha}_{\text{pool}}\right)}\right)$, where recall that $\bar{\alpha}_{\text{pool}} = \min\{\alpha_{\text{pool}}, 1-\alpha_{\text{pool}}\}$, as defined in the statement of the Lemma. It is easy to check that $0 < t(\varepsilon) \leq \alpha_{\text{pool}} \cdot \varepsilon'$ where note from above that $\varepsilon' = \frac{-\varepsilon \log \alpha_{\text{pool}}}{1-\varepsilon \log \alpha_{\text{pool}}}$. Then using Lemma A.4, we get that

$$\mathbb{P}\Big[\big|\boldsymbol{F}_0^+ - \alpha_{\text{pool}}\big| \leq \varepsilon' \alpha_{\text{pool}}\Big] \geq \mathbb{P}\Big[\big|\boldsymbol{F}_0^+ - \alpha_{\text{pool}}\big| \leq t(\varepsilon)\Big] \geq 1 - 2\exp\big(-2m\ell \cdot t^2(\varepsilon)\big),$$

where for simplicity of notation, we denote $(t(\varepsilon))^2$ as $t^2(\varepsilon)$. Then, using equation (A.2) it follows that

$$\begin{aligned}\mathbb{P}\Big[\big|-\log \boldsymbol{F}_0^+ - (-\log \alpha_{\text{pool}})\big| \leq \varepsilon \cdot (-\log \alpha_{\text{pool}})\Big] &\geq \mathbb{P}\Big[\big|\boldsymbol{F}_0^+ - \alpha_{\text{pool}}\big| \leq \varepsilon' \alpha_{\text{pool}}\Big] \\ &\geq 1 - 2\exp\big(-2m\ell \cdot t^2(\varepsilon)\big)\end{aligned} \tag{A.3}$$

A similar sequence of arguments (using the random variable $1-\boldsymbol{F}_0^+$) shows that for $\varepsilon'' = \frac{-\varepsilon \log(1-\alpha_{\text{pool}})}{1-\varepsilon \log(1-\alpha_{\text{pool}})}$ and observing that $0 < t(\varepsilon) \leq (1-\alpha_{\text{pool}}) \cdot \varepsilon''$, we get

$$\begin{aligned}&\mathbb{P}\Big[\big|-\log(1-\boldsymbol{F}_0^+) - (-\log(1-\alpha_{\text{pool}}))\big| \leq \varepsilon \cdot (-\log(1-\alpha_{\text{pool}}))\Big] \\ &\geq \mathbb{P}\Big[\big|\boldsymbol{F}_0^+ - \alpha_{\text{pool}}\big| \leq \varepsilon'' \cdot (1-\alpha_{\text{pool}})\Big] \geq 1 - 2\exp\big(-2m\ell \cdot t^2(\varepsilon)\big)\end{aligned} \tag{A.4}$$

For ease of notation in the remainder of the proof, denote the embedding score $\boldsymbol{V}_i = \frac{\boldsymbol{N}_i}{\boldsymbol{D}_i}$ to specify the numerator and denominator terms.

Concentration of $\boldsymbol{N}_i$. Let us begin with the numerator, $\boldsymbol{N}_i = -\boldsymbol{F}_i^+ \log \boldsymbol{F}_0^+ - (1-\boldsymbol{F}_i^+)\log(1-\boldsymbol{F}_0^+)$. Consider the first term: $\boldsymbol{F}_i^+ \cdot (-\log \boldsymbol{F}_0^+)$ and note that $\mathbb{E}[\boldsymbol{F}_i^+] = \alpha_{z_i}$, $\mathbb{E}[\boldsymbol{F}_0^+] = \alpha_{\text{pool}}$. Then using Lemma A.2 with $\boldsymbol{X} = \boldsymbol{F}_i^+, \boldsymbol{Y} = \boldsymbol{F}_0^+$ and denoting $A_1 = \hat{c}_1 \stackrel{\text{def}}{=}$

$\alpha_{z_i} \cdot (-\log \alpha_{\text{pool}})$, we get:

$$\begin{aligned}
&\mathbb{P}\Big[\big|\boldsymbol{F}_i^+ \cdot (-\log \boldsymbol{F}_0^+) - \hat{c}_1\big| > \varepsilon \hat{c}_1\Big] \\
&\leq \mathbb{P}\Big[\big|\boldsymbol{F}_i^+ - \alpha_{z_i}\big| > \frac{\varepsilon}{3}\alpha_{z_i}\Big] + \mathbb{P}\Big[\big|-\log \boldsymbol{F}_0^+ - (-\log \alpha_{\text{pool}})\big| > \frac{\varepsilon}{3} \cdot (-\log \alpha_{\text{pool}})\Big] \\
&\leq 2\exp\Big(-2\ell\frac{\varepsilon^2\alpha_{z_i}^2}{9}\Big) + 2\exp\Big(-2m\ell \cdot t^2(\varepsilon/3)\Big) \quad \{\text{using equations (A.1) and (A.3)}\} \\
&\leq 2\exp\Big(-2\ell\frac{\varepsilon^2\alpha_{\min}^2}{9}\Big) + 2\exp\Big(-2m\ell \cdot t^2(\varepsilon/3)\Big) \quad \{\text{since } \alpha_{z_i} \geq \alpha_{\min}\} \qquad \text{(A.5)}
\end{aligned}$$

Similarly for the second term, observe that $\mathbb{E}[1 - \boldsymbol{F}_i^+] = 1 - \alpha_{z_i}$, $\mathbb{E}[1 - \boldsymbol{F}_0^+] = 1 - \alpha_{\text{pool}}$. Therefore, choosing $\boldsymbol{X} = (1 - \boldsymbol{F}_i^+), \boldsymbol{Y} = 1 - \boldsymbol{F}_0^+$ and denoting $A_1 = \hat{c}_2 \overset{\text{def}}{=} (1 - \alpha_{z_i}) \cdot \Big(-\log(1 - \alpha_{\text{pool}})\Big)$ in Lemma A.2, we get:

$$\begin{aligned}
&\mathbb{P}\Big[\Big|(1 - \boldsymbol{F}_i^+) \cdot \Big(-\log(1 - \boldsymbol{F}_0^+)\Big) - \hat{c}_2\Big| > \varepsilon \hat{c}_2\Big] \\
&\leq \mathbb{P}\Big[\big|(1 - \boldsymbol{F}_i^+) - (1 - \alpha_{z_i})\big| > \frac{\varepsilon}{3} \cdot (1 - \alpha_{z_i})\Big] \\
&+ \mathbb{P}\Big[\big|-\log(1 - \boldsymbol{F}_0^+) - (-\log(1 - \alpha_{\text{pool}}))\big| > \frac{\varepsilon}{3} \cdot (-\log(1 - \alpha_{\text{pool}}))\Big] \\
&\leq 2\exp\Big(-2\ell\frac{\varepsilon^2(1 - \alpha_{z_i})^2}{9}\Big) + 2\exp\Big(-2m\ell \cdot t^2(\varepsilon/3)\Big) \quad \{\text{using equations (A.1) and (A.4)}\} \\
&\leq 2\exp\Big(-2\ell\frac{\varepsilon^2\alpha_{\min}^2}{9}\Big) + 2\exp\Big(-2m\ell \cdot t^2(\varepsilon/3)\Big) \quad \{\text{since } (1 - \alpha_{z_i}) \geq \alpha_{\min}\} \qquad \text{(A.6)}
\end{aligned}$$

Combining the above two, choosing $\boldsymbol{X}_1 = \boldsymbol{F}_i^+ \cdot (-\log \boldsymbol{F}_0^+)$, $\boldsymbol{X}_2 = (1 - \boldsymbol{F}_i^+) \cdot (-\log(1 - \boldsymbol{F}_0^+))$, $a_1 = \hat{c}_1$ and $a_2 = \hat{c}_2$ in Lemma A.1 (iii), we get:

$$\begin{aligned}
&\mathbb{P}\Big[|\boldsymbol{N}_i - (\hat{c}_1 + \hat{c}_2)| > \frac{\varepsilon}{3} \cdot (\hat{c}_1 + \hat{c}_2)\Big] \\
&\leq \mathbb{P}\Big[\big|\boldsymbol{F}_i^+ \cdot (-\log \boldsymbol{F}_0^+) - \hat{c}_1\big| > \frac{\varepsilon}{3}\hat{c}_1\Big] + \mathbb{P}\Big[\Big|(1 - \boldsymbol{F}_i^+) \cdot \Big(-\log(1 - \boldsymbol{F}_0^+)\Big) - \hat{c}_2\Big| > \frac{\varepsilon}{3}\hat{c}_2\Big] \\
&\leq 4\exp\Big(-2\ell\frac{\varepsilon^2\alpha_{\min}^2}{81}\Big) + 4\exp\Big(-2m\ell \cdot t^2(\varepsilon/9)\Big) \\
&\{\text{using equations (A.5) and (A.6)}\} \qquad \text{(A.7)}
\end{aligned}$$

Concentration of $\boldsymbol{D}_i$. Moving on to the denominator, $\boldsymbol{D}_i = \boldsymbol{F}_0^+ \cdot (-\log \boldsymbol{F}_0^+) + (1 - \boldsymbol{F}_0^+) \cdot (-\log(1 - \boldsymbol{F}_0^+))$. Focusing on the first term, $\boldsymbol{F}_0^+ \cdot (-\log \boldsymbol{F}_0^+)$, observe that $\mathbb{E}[\boldsymbol{F}_0^+] = \alpha_{\text{pool}}$.

Again using Lemma A.2 with $\boldsymbol{X} = \boldsymbol{F}_0^+$ and denoting $A_2 = \hat{b}_1 \stackrel{\text{def}}{=} \alpha_{\text{pool}} \cdot (-\log \alpha_{\text{pool}})$ we get,

$$
\begin{aligned}
&\mathbb{P}\left[\left|\boldsymbol{F}_0^+ \cdot (-\log \boldsymbol{F}_0^+) - \hat{b_1}\right| > \varepsilon \hat{b_1}\right] \\
&\leq \mathbb{P}\left[\left|\boldsymbol{F}_0^+ - \alpha_{\text{pool}}\right| > \frac{\varepsilon}{3}\alpha_{\text{pool}}\right] + \mathbb{P}\left[\left|-\log \boldsymbol{F}_0^+ - (-\log \alpha_{\text{pool}})\right| > \frac{\varepsilon}{3} \cdot (-\log \alpha_{\text{pool}})\right] \\
&\leq 2\exp\left(-2m\ell\frac{\varepsilon^2}{9}\alpha_{\text{pool}}^2\right) + 2\exp\left(-2m\ell \cdot t^2(\varepsilon/3)\right) \quad \{\text{using Lemma A.4 and eq. (A.3)}\} \\
&\leq 2\exp\left(-2m\ell\frac{\varepsilon^2}{9}\bar{\alpha}_{\text{pool}}^2\right) + 2\exp\left(-2m\ell \cdot t^2(\varepsilon/3)\right) \quad \{\text{since } \alpha_{\text{pool}} \geq \bar{\alpha}_{\text{pool}}\}
\end{aligned}
$$

Similarly, for the second term choosing $\boldsymbol{X} = (1 - \boldsymbol{F}_0^+)$ and denoting $A_2 = \hat{b}_2 \stackrel{\text{def}}{=} (1 - \alpha_{\text{pool}}) \cdot (-\log(1 - \alpha_{\text{pool}}))$ in Lemma A.2 we get:

$$
\begin{aligned}
&\mathbb{P}\left[\left|(1 - \boldsymbol{F}_0^+) \cdot (-\log(1 - \boldsymbol{F}_0^+)) - \hat{b_2}\right| > \varepsilon \hat{b_2}\right] \\
&\leq \mathbb{P}\left[\left|(1 - \boldsymbol{F}_0^+) - (1 - \alpha_{\text{pool}})\right| > \frac{\varepsilon}{3} \cdot (1 - \alpha_{\text{pool}})\right] + \\
&\mathbb{P}\left[\left|-\log(1 - \boldsymbol{F}_0^+) - (-\log(1 - \alpha_{\text{pool}}))\right| > \frac{\varepsilon}{3} \cdot (-\log(1 - \alpha_{\text{pool}}))\right] \\
&= \mathbb{P}\left[\left|\boldsymbol{F}_0^+ - \alpha_{\text{pool}}\right| > \frac{\varepsilon}{3} \cdot (1 - \alpha_{\text{pool}})\right] + \\
&\mathbb{P}\left[\left|-\log(1 - \boldsymbol{F}_0^+) - (-\log(1 - \alpha_{\text{pool}}))\right| > \frac{\varepsilon}{3} \cdot (-\log(1 - \alpha_{\text{pool}}))\right] \\
&\leq 2\exp\left(-2m\ell\frac{\varepsilon^2}{9}(1 - \alpha_{\text{pool}})^2\right) + 2\exp\left(-2m\ell \cdot t^2(\varepsilon/3)\right) \quad \{\text{using Lemma A.4 and eq. (A.4)}\} \\
&\leq 2\exp\left(-2m\ell\frac{\varepsilon^2}{9}\bar{\alpha}_{\text{pool}}^2\right) + 2\exp\left(-2m\ell \cdot t^2(\varepsilon/3)\right) \quad \{\text{since } (1 - \alpha_{\text{pool}}) \geq \bar{\alpha}_{\text{pool}}\}
\end{aligned}
$$

Combining the above two, choosing $\boldsymbol{X}_1 = \boldsymbol{F}_0^+ \cdot (-\log \boldsymbol{F}_0^+)$, $\boldsymbol{X}_2 = (1 - \boldsymbol{F}_0^+) \cdot (-\log(1 - \boldsymbol{F}_0^+))$, $a_1 = \hat{b}_1$ and $a_2 = \hat{b}_2$ in Lemma A.1 (iii), we get:

$$
\begin{aligned}
&\mathbb{P}\left[\left|\boldsymbol{D}_i - (\hat{b}_1 + \hat{b_2})\right| > \frac{\varepsilon}{3}(\hat{b}_1 + \hat{b}_2)\right] \\
&\leq \mathbb{P}\left[\left|\boldsymbol{F}_0^+ \cdot (-\log \boldsymbol{F}_0^+) - \hat{b}_1\right| > \frac{\varepsilon}{3}\hat{b}_1\right] + \mathbb{P}\left[\left|(1 - \boldsymbol{F}_0^+) \cdot \left(-\log(1 - \boldsymbol{F}_0^+)\right) - \hat{b}_2\right| > \frac{\varepsilon}{3}\hat{b}_2\right] \\
&\leq 4\exp\left(-2m\ell\frac{\varepsilon^2}{81}\bar{\alpha}_{\text{pool}}^2\right) + 4\exp\left(-2m\ell \cdot t^2(\varepsilon/9)\right) \qquad \text{(A.8)}
\end{aligned}
$$

Concentration of $\boldsymbol{V}_i$. Now that we have expressions for the concentration of the numerator and denominator, we can discuss the concentration of the embedding score $\boldsymbol{V}_i$.

Choosing $\boldsymbol{X}_i = \boldsymbol{N}_i$, $\boldsymbol{Y} = \boldsymbol{D}_i$, $a_i = \hat{c_1} + \hat{c_2}$, $b = \hat{b_1} + \hat{b_2}$ in Lemma A.1 (i), we get the required concentration bound for the unidimensional embedding of customer i:

$$
\begin{aligned}
&\mathbb{P}\left[\left|\boldsymbol{V}_i - \frac{\hat{c_1}+\hat{c_2}}{\hat{b_1}+\hat{b_2}}\right| > \varepsilon\frac{\hat{c_1}+\hat{c_2}}{\hat{b_1}+\hat{b_2}}\right] \\
&= \mathbb{P}\left[\left|\frac{\boldsymbol{N}_i}{\boldsymbol{D}_i} - \frac{\hat{c_1}+\hat{c_2}}{\hat{b_1}+\hat{b_2}}\right| > \varepsilon\frac{\hat{c_1}+\hat{c_2}}{\hat{b_1}+\hat{b_2}}\right] \\
&\leq \mathbb{P}\left[|\boldsymbol{N}_i - (\hat{c_1}+\hat{c_2})| > \frac{\varepsilon}{3}\cdot(\hat{c_1}+\hat{c_2})\right] + \mathbb{P}\left[\left|\boldsymbol{D}_i - (\hat{b_1}+\hat{b_2})\right| > \frac{\varepsilon}{3}\cdot(\hat{b_1}+\hat{b_2})\right] \\
&\leq 4\exp\left(-2\ell\frac{\varepsilon^2\alpha_{\min}^2}{81}\right) + 4\exp\left(-2m\ell\frac{\varepsilon^2}{81}\bar{\alpha}_{\text{pool}}^2\right) + 8\exp\left(-2m\ell\cdot t^2(\varepsilon/9)\right) \\
&\{\text{from equations (A.7) and (A.8)}\} \\
&\leq 4\exp\left(-2\ell\frac{\varepsilon^2\alpha_{\min}^2}{81}\right) + 12\exp\left(-2m\ell\cdot t^2(\varepsilon/9)\right) \\
&\left(\text{since } \frac{\log^2(1-\bar{\alpha}_{\text{pool}})}{(1-\log(1-\bar{\alpha}_{\text{pool}}))^2} < 1\right)
\end{aligned}
$$

Finally, note that $\hat{c_1} + \hat{c_2} = H(\alpha_{z_i}, \alpha_{\text{pool}})$, the cross-entropy between the distributions $\text{Ber}(\alpha_{z_i})$ and $\text{Ber}(\alpha_{\text{pool}})$, and $\hat{b_1} + \hat{b_2} = H(\alpha_{\text{pool}})$, the binary entropy function evaluated at α_{pool}. The result then follows. □

Proof of Theorem 1.2. The result follows directly from the fact (proved above) that embedding scores of customers in segment k concentrate around the ratio $\frac{H(\alpha_k, \alpha_{\text{pool}})}{H(\alpha_{\text{pool}})}$; refer to the discussion after Lemma 1.1 in the main text.

A.1.3 Proof of Theorem 1.3

We begin by proving some useful lemmas. All notations are as stated in the main text, unless otherwise introduced.

Lemma A.5. *Let k_1, k_2 be two arbitrary segments. Then for customer i, we have*

$$
\frac{|\boldsymbol{V}_i - H_{k_1}|}{H_{k_1}} \leq \frac{|H_{k_1} - H_{k_2}|}{2\cdot\max(H_{k_1}, H_{k_2})} \implies \frac{|\boldsymbol{V}_i - H_{k_1}|}{H_{k_1}} \leq \frac{|\boldsymbol{V}_i - H_{k_2}|}{H_{k_2}}
$$

Proof. Consider the following:

$$\begin{aligned}\frac{|H_{k_1} - H_{k_2}|}{\max(H_{k_1}, H_{k_2})} &= \frac{|(H_{k_1} - \boldsymbol{V}_i) + (\boldsymbol{V}_i - H_{k_2})|}{\max(H_{k_1}, H_{k_2})} \\ &\leq \frac{|\boldsymbol{V}_i - H_{k_1}|}{\max(H_{k_1}, H_{k_2})} + \frac{|\boldsymbol{V}_i - H_{k_2}|}{\max(H_{k_1}, H_{k_2})} \\ &\text{(using triangle inequality)} \\ &\leq \frac{|\boldsymbol{V}_i - H_{k_1}|}{H_{k_1}} + \frac{|\boldsymbol{V}_i - H_{k_2}|}{H_{k_2}} \\ &\leq \frac{|H_{k_1} - H_{k_2}|}{2 \cdot \max(H_{k_1}, H_{k_2})} + \frac{|\boldsymbol{V}_i - H_{k_2}|}{H_{k_2}} \\ &\text{(follows from the hypothesis of the Lemma)}\end{aligned}$$

Therefore we have that

$$\frac{|\boldsymbol{V}_i - H_{k_2}|}{H_{k_2}} \geq \frac{|H_{k_1} - H_{k_2}|}{2 \cdot \max(H_{k_1}, H_{k_2})} \geq \frac{|\boldsymbol{V}_i - H_{k_1}|}{H_{k_1}}$$

□

Lemma A.6. *Consider the constant* Λ *defined in Theorem 1.3:*

$$\Lambda = \frac{\left|\log \frac{\alpha_{\text{pool}}}{1-\alpha_{\text{pool}}}\right| \cdot \lambda}{2\left|\log \alpha_{\min}\right|}$$

Then, it follows that $\Lambda \leq \min_{k \neq k'} \frac{|H_k - H_{k'}|}{2 \cdot \max(H_k, H_{k'})} < 1$.

Proof. Recall that $H_k = \frac{H(\alpha_k, \alpha_{\text{pool}})}{H(\alpha_{\text{pool}})}$. Then, for any two segments $k \neq k'$, define:

$$\Lambda_{kk'} \stackrel{\text{def}}{=} \frac{|H_k - H_{k'}|}{2 \cdot \max(H_k, H_{k'})} = \frac{|H(\alpha_k, \alpha_{\text{pool}}) - H(\alpha_{k'}, \alpha_{\text{pool}})|}{2 \cdot \max\{H(\alpha_k, \alpha_{\text{pool}}), H(\alpha_{k'}, \alpha_{\text{pool}})\}}$$

Next, observe that $H(\alpha_k, \alpha_{\text{pool}}) = -\alpha_k \log \frac{\alpha_{\text{pool}}}{1-\alpha_{\text{pool}}} - \log(1 - \alpha_{\text{pool}})$, so that

$$|H(\alpha_k, \alpha_{\text{pool}}) - H(\alpha_{k'}, \alpha_{\text{pool}})| = \left|\log \frac{\alpha_{\text{pool}}}{1 - \alpha_{\text{pool}}}\right| |\alpha_k - \alpha_{k'}|$$

Now suppose $\alpha_{\text{pool}} > \frac{1}{2}$, this means that $H(\alpha_k, \alpha_{\text{pool}})$ is decreasing with α_k so that we have

$$\max\{H(\alpha_k, \alpha_{\text{pool}}), H(\alpha_{k'}, \alpha_{\text{pool}})\} \leq H(\alpha_{\min}, \alpha_{\text{pool}}) \leq H(\alpha_{\min}, 1 - \alpha_{\min}) \leq -\log \alpha_{\min}$$

where the second inequality follows from the fact that $\alpha_{\text{pool}} \leq 1 - \alpha_{\min}$ and the last inequality from the fact that $-\log(1-\alpha_{\min}) \leq -\log \alpha_{\min}$. Similarly, when $\alpha_{\text{pool}} < \frac{1}{2}$, we have

$$\begin{aligned}\max\{H(\alpha_k, \alpha_{\text{pool}}), H(\alpha_{k'}, \alpha_{\text{pool}})\} &\leq H(1-\alpha_{\min}, \alpha_{\text{pool}}) \\ &\leq H(1-\alpha_{\min}, \alpha_{\min}) = H(\alpha_{\min}, 1-\alpha_{\min}) \leq -\log \alpha_{\min}\end{aligned}$$

where the second inequality is true because $\alpha_{\text{pool}} \geq \alpha_{\min}$.

Combining the above observations and using the fact that $|\alpha_k - \alpha_{k'}| \geq \lambda$ for all $k \neq k'$, we get $\Lambda_{kk'} \geq \Lambda$ for all $k \neq k'$. Further, observe that $\Lambda_{kk'} < 1$ because $H_k > 0$ for all $1 \leq k \leq K$. Therefore, $\Lambda \leq \min_{k \neq k'} \Lambda_{kk'} < 1$ and the claim follows. □

Lemma A.7. *Consider customer i and suppose we have the following:*

$$\frac{|\boldsymbol{V}_i - H_{z_i}|}{H_{z_i}} \leq \frac{|H_{z_i} - H_{k'}|}{2 \cdot \max(H_{z_i}, H_{k'})} \quad \forall\, k' \neq z_i$$

Then it follows that $\hat{\boldsymbol{I}}(i) = z_i$, i.e the NN classifier $\hat{\boldsymbol{I}}(\cdot)$ correctly classifies customer i. Conversely, it follows that

$$\mathbb{P}\Big[\hat{\boldsymbol{I}}(i) \neq z_i\Big] \leq \mathbb{P}\Big[\,|\boldsymbol{V}_i - H_{z_i}| > \Lambda \cdot H_{z_i}\Big].$$

Proof. Using Lemma A.5 we obtain that $\frac{|\boldsymbol{V}_i - H_{z_i}|}{H_{z_i}} \leq \frac{|\boldsymbol{V}_i - H_{k'}|}{H_{k'}}$ for all $k' \neq z_i$. This means that $\arg\min_{k \in [K]} \frac{|\boldsymbol{V}_i - H_k|}{H_k} = z_i$. For the second part of the claim, observe that if $\hat{\boldsymbol{I}}(i) \neq z_i$, then by Lemma A.6 it follows that there exists some $k \neq z_i$ such that $\frac{|\boldsymbol{V}_i - H_{z_i}|}{H_{z_i}} > \frac{|H_{z_i} - H_k|}{2 \cdot \max(H_{z_i}, H_k)} \geq \Lambda$. In other words,

$$\hat{\boldsymbol{I}}(i) \neq z_i \implies |\boldsymbol{V}_i - H_{z_i}| > \Lambda \cdot H_{z_i}$$

and the claim follows. □

We now have all the ingredients for the proof. First, observe that the probability customer i is misclassified by the nearest-neighbor classifier $\hat{\boldsymbol{I}}(\cdot)$ is given by:

$$\begin{aligned}\mathbb{P}\Big[\hat{\boldsymbol{I}}(i) \neq z_i\Big] &\leq \mathbb{P}\Big[\,|\boldsymbol{V}_i - H_{z_i}| > \Lambda \cdot H_{z_i}\Big] \quad \text{(using Lemma A.7)} \\ &\leq 4\exp\left(-2\ell\frac{\Lambda^2\alpha_{\min}^2}{81}\right) + 12\exp\left(\frac{-2m \cdot \ell \cdot \Lambda^2 \bar{\alpha}_{\text{pool}}^2 \log^2(1-\bar{\alpha}_{\text{pool}})}{81\big(1-\log(1-\bar{\alpha}_{\text{pool}})\big)^2}\right) \\ &\qquad \text{(using result of Lemma 1.1)} \qquad \text{(A.9)}\end{aligned}$$

Now given any $0 < \delta < 1$, suppose that the number of observations from each customer satisfies:

$$\ell \geq \frac{648}{\lambda^2} \cdot \left(\frac{\log \alpha_{\min}}{\log(1-\alpha_{\min}) \cdot \alpha_{\min}}\right)^2 \cdot \frac{1}{\log^2 \frac{\alpha_{\text{pool}}}{1-\alpha_{\text{pool}}}} \cdot \log(16/\delta)$$

Then, starting from equation (A.9), it follows that

$$\begin{aligned}
\mathbb{P}\Big[\hat{\boldsymbol{I}}(i) \neq z_i\Big] &\leq 4\exp\Big(-2\ell\frac{\Lambda^2\alpha_{\min}^2}{81}\Big) + 12\exp\left(\frac{-2m\cdot\ell\cdot\Lambda^2\bar{\alpha}_{\text{pool}}^2\log^2(1-\bar{\alpha}_{\text{pool}})}{81\big(1-\log(1-\bar{\alpha}_{\text{pool}})\big)^2}\right) \\
&\leq 4\exp\Big(-2\ell\frac{\Lambda^2\alpha_{\min}^2}{81}\Big) + 12\exp\left(\frac{-2m\cdot\ell\cdot\Lambda^2\alpha_{\min}^2\log^2(1-\alpha_{\min})}{81\big(1-\log(1-\alpha_{\min})\big)^2}\right) \\
&\quad \text{(since } \bar{\alpha}_{\text{pool}} \geq \alpha_{\min}\text{)} \\
&\leq 4\exp\Big(-2\ell\frac{\Lambda^2\alpha_{\min}^2\log^2(1-\alpha_{\min})}{81\,(1-\log(1-\alpha_{\min}))^2}\Big) + 12\exp\left(\frac{-2m\cdot\ell\cdot\Lambda^2\alpha_{\min}^2\log^2(1-\alpha_{\min})}{81\big(1-\log(1-\alpha_{\min})\big)^2}\right) \\
&\quad \Big(\text{since } \log^2(1-\alpha_{\min}) < (1-\log(1-\alpha_{\min}))^2\Big) \\
&\leq 16\exp\Big(-2\ell\frac{\Lambda^2\alpha_{\min}^2\log^2(1-\alpha_{\min})}{81\,(1-\log(1-\alpha_{\min}))^2}\Big) \\
&\quad \text{(since } m \geq 1\text{)} \\
&= 16\exp\Big(-\ell\frac{\lambda^2\alpha_{\min}^2\log^2\frac{\alpha_{\text{pool}}}{1-\alpha_{\text{pool}}}\log^2(1-\alpha_{\min})}{162\cdot\log^2\alpha_{\min}\cdot(1-\log(1-\alpha_{\min}))^2}\Big) \\
&\quad \text{(substituting the value of } \Lambda\text{)} \\
&\leq 16\exp\Big(-\ell\frac{\lambda^2\alpha_{\min}^2\log^2\frac{\alpha_{\text{pool}}}{1-\alpha_{\text{pool}}}\log^2(1-\alpha_{\min})}{648\cdot\log^2\alpha_{\min}}\Big) \\
&\quad \Big(\text{since } \alpha_{\min} < \frac{1}{2} \implies 1-\log(1-\alpha_{\min}) < 2\Big) \\
&\leq \delta \\
&\quad \Big(\text{using the bound on } \ell\Big)
\end{aligned}$$

Next, suppose that $m\cdot\frac{\log^2(1-\bar{\alpha}_{\text{pool}})}{\left(1-\log(1-\bar{\alpha}_{\text{pool}})\right)^2} \geq 1$, and observe that $\bar{\alpha}_{\text{pool}} \geq \alpha_{\min}$. Then we get,

$$\begin{aligned}
\mathbb{P}\Big[\hat{\boldsymbol{I}}(i) \neq z_i\Big] &\leq \mathbb{P}\Big[\,|\boldsymbol{V}_i - H_{z_i}| > \Lambda\cdot H_{z_i}\,\Big] \quad \text{(using Lemma A.7)} \\
&\leq 4\exp\Big(-2\ell\frac{\Lambda^2\alpha_{\min}^2}{81}\Big) + 12\exp\left(\frac{-2m\cdot\ell\cdot\Lambda^2\bar{\alpha}_{\text{pool}}^2\log^2(1-\bar{\alpha}_{\text{pool}})}{81\big(1-\log(1-\bar{\alpha}_{\text{pool}})\big)^2}\right) \\
&\quad \text{(using result of Lemma 1.1)} \\
&\leq 16\exp\Big(-2\ell\frac{\Lambda^2\alpha_{\min}^2}{81}\Big) \\
&\quad \text{(using the lower bounds on } m \text{ and } \bar{\alpha}_{\text{pool}}\text{)}
\end{aligned}$$

Substituting $\ell = \log n$ above, we get the desired result.

A.1.4 Proofs of Section 1.3.2

Recall that segment k is characterized by B-dimensional vector $\boldsymbol{\alpha}_k$ such that α_{kb} represents the probability of liking any item $j \in \mathcal{I}_b$. Let $\boldsymbol{X}^+_{ib}$ denote the number of likes given by customer i for items in category b, i.e. $\boldsymbol{X}^+_{ib} = \sum_{j \in N_b(i)} \mathbf{1}[\boldsymbol{X}_{ij} = +1]$, where $N_b(i) \subset \mathcal{I}_b$ denotes the collection of items in category b rated by customer i.

To calculate the embedding scores, we first need to compute the pooled estimate for each category. Since the underlying LC-IND-CAT model is parameterized by a vector of length B for each segment, the pooled estimate is given by $\overrightarrow{\boldsymbol{F}^+_0} = (\boldsymbol{F}^+_{01}, \boldsymbol{F}^+_{02}, \ldots, \boldsymbol{F}^+_{0B})$ with:

$$\boldsymbol{F}^+_{0b} \stackrel{\text{def}}{=} \frac{\sum_{i'=1}^{m} \boldsymbol{X}^+_{i'b}}{m \cdot \ell_b} \quad \forall\, b \in [B]$$

where ℓ_b is the number of items in category b that each customer rates. Also, let us denote the fraction of likes given by customer i for category b items as $\boldsymbol{F}^+_{ib} \stackrel{\text{def}}{=} \frac{\boldsymbol{X}^+_{ib}}{\ell_b}$. We first establish a lemma that will be useful in the proof:

Lemma A.8. *Given any $t > 0$, for each category $b \in [B]$, the following facts are true:*

$$\begin{aligned}
&\text{(i)} \quad \boldsymbol{X}^+_{ib} \sim \text{Bin}(\ell_b, \alpha_{z_i b}) \\
&\text{(ii)} \quad \mathbb{E}[\boldsymbol{F}^+_{0b}] = \alpha_{b,\text{pool}} \\
&\text{(iii)} \quad \mathbb{P}\Big[\, \big|\boldsymbol{F}^+_{0b} - \alpha_{b,\text{pool}}\big| \geq t \,\Big] \leq 2\exp\big(-2m \cdot \ell_b \cdot t^2\big)
\end{aligned}$$

Proof. Lets begin with part (i). Observe that $\boldsymbol{X}^+_{ib} = \sum\limits_{j \in N_b(i)} \mathbf{1}[\boldsymbol{X}_{ij} = +1]$. Based on the LC-IND-CAT model, we have that $\mathbf{1}[\boldsymbol{X}_{ij} = +1]$ are i.i.d. such that $\mathbb{P}[\boldsymbol{X}_{ij} = +1] = \alpha_{z_i b}$. The claim follows from the definition of $\boldsymbol{X}^+_{ib}$.

For part (ii) observe that,

$$\mathbb{E}[\boldsymbol{F}^+_{0b}] = \frac{\sum_{i'=1}^{m} \mathbb{E}[\boldsymbol{X}^+_{i'b}]}{m \cdot \ell_b} = \frac{\sum_{i'=1}^{m} \ell_b \alpha_{z_{i'} b}}{m \cdot \ell_b} = \frac{\sum_{k'=1}^{K} (q_{k'} m) \cdot (\ell_b \alpha_{k'b})}{m \cdot \ell_b} = \sum_{k'=1}^{K} q_{k'} \alpha_{k'b} = \alpha_{b,\text{pool}}$$

where the third equality follows from the fact that proportion $q_{k'}$ of the customer population belongs to segment k'.

For part (iii), observe that $\boldsymbol{F}^+_{0b}$ can be written as:

$$\boldsymbol{F}^+_{0b} = \frac{\sum\limits_{i'=1}^{m} \sum_{j \in N_b(i)} \mathbf{1}[\boldsymbol{X}_{i'j} = +1]}{m \cdot \ell_b}$$

In other words, $\boldsymbol{F}^+_{0b}$ is an average of $m \cdot \ell_b$ random variables, which are independent under the LC-IND-CAT model (since ratings for items within the same category are independent and

the observations of different customers are generated independently). Then using Hoeffding's inequality we can show, for any $t > 0$:

$$\mathbb{P}\Big[\,\big|\boldsymbol{F}_{0b}^{+} - \alpha_{b,\text{pool}}\big| \geq t\,\Big] \leq 2\exp\big(-2m\cdot\ell_b\cdot t^2\big)$$

□

A.1.5 Proof of Lemma 1.4

For customer i that belongs to segment k, the embedding vector computed by our algorithm, $\overrightarrow{\boldsymbol{V}}_i = (\boldsymbol{V}_{i1}, \boldsymbol{V}_{i2}, \cdots, \boldsymbol{V}_{iB})$, is such that:

$$\begin{aligned}
\boldsymbol{V}_{ib} &= \frac{-\sum_{j\in N_b(i)}\big(\mathbf{1}[\boldsymbol{X}_{ij}=+1]\log\boldsymbol{F}_{0b}^{+} + \mathbf{1}[\boldsymbol{X}_{ij}=-1]\log(1-\boldsymbol{F}_{0b}^{+})\big)}{-\sum_{j\in N_b(i)}\big(\boldsymbol{F}_{0b}^{+}\log\boldsymbol{F}_{0b}^{+} + (1-\boldsymbol{F}_{0b}^{+})\log(1-\boldsymbol{F}_{0b}^{+})\big)} \\
&= \frac{-\big(\sum_{j\in N_b(i)}\mathbf{1}[\boldsymbol{X}_{ij}=+1]\big)\log\boldsymbol{F}_{0b}^{+} - \big(\sum_{j\in N_b(i)}\mathbf{1}[\boldsymbol{X}_{ij}=-1]\big)\log(1-\boldsymbol{F}_{0b}^{+})}{-\big(\boldsymbol{F}_{0b}^{+}\log\boldsymbol{F}_{0b}^{+} + (1-\boldsymbol{F}_{0b}^{+})\log(1-\boldsymbol{F}_{0b}^{+})\big)\cdot\ell_b} \\
&= \frac{-\big(\frac{\boldsymbol{X}_{ib}^{+}}{\ell_b}\big)\log\boldsymbol{F}_{0b}^{+} - \big(1-\frac{\boldsymbol{X}_{ib}^{+}}{\ell_b}\big)\log(1-\boldsymbol{F}_{0b}^{+})}{-\big(\boldsymbol{F}_{0b}^{+}\log\boldsymbol{F}_{0b}^{+} + (1-\boldsymbol{F}_{0b}^{+})\log(1-\boldsymbol{F}_{0b}^{+})\big)} \\
&= \frac{-\boldsymbol{F}_{ib}^{+}\log\boldsymbol{F}_{0b}^{+} - \big(1-\boldsymbol{F}_{ib}^{+}\big)\log(1-\boldsymbol{F}_{0b}^{+})}{-\big(\boldsymbol{F}_{0b}^{+}\log\boldsymbol{F}_{0b}^{+} + (1-\boldsymbol{F}_{0b}^{+})\log(1-\boldsymbol{F}_{0b}^{+})\big)}
\end{aligned}$$

Observe that the exact sequence of arguments given in the proof of Lemma 1.1 earlier can be repeated, for each item category b separately. Then, it follows that, given any $0 < \varepsilon < 1$, and for each $b \in [B]$:

$$\begin{aligned}
\mathbb{P}\Big[\,\Big|\boldsymbol{V}_{ib} - \frac{H(\alpha_{z_i b},\alpha_{b,\text{pool}})}{H(\alpha_{b,\text{pool}})}\Big| > \varepsilon\frac{H(\alpha_{z_i b},\alpha_{b,\text{pool}})}{H(\alpha_{b,\text{pool}})}\Big] \\
\leq 4\exp\left(-2\ell_b\frac{\varepsilon^2\alpha_{\min}^2}{81}\right) + 12\exp\Big(-2m\cdot\ell_b\cdot t_b^2(\varepsilon/9)\Big) \qquad \text{(A.10)}
\end{aligned}$$

where $t_b(\varepsilon) \stackrel{\text{def}}{=} \varepsilon\cdot\left(\frac{-\bar{\alpha}_{b,\text{pool}}\log(1-\bar{\alpha}_{b,\text{pool}})}{1-\log(1-\bar{\alpha}_{b,\text{pool}})}\right)$ and $\bar{\alpha}_{b,\text{pool}} = \min\{\alpha_{b,\text{pool}}, 1-\alpha_{b,\text{pool}}\}$.

Then, we consider the concentration of the vector $\overrightarrow{\boldsymbol{V}}_i$. Recall the B-dimensional vector $\boldsymbol{H}_k = (H_{k1}, H_{k2}, \cdots, H_{kB})$ for each $k \in [K]$, defined in the main text, with $H_{kb} = \frac{H(\alpha_{kb},\alpha_{b,\text{pool}})}{H(\alpha_{b,\text{pool}})}$.

Then, using Lemma A.1(iv) it follows that:

$$\begin{aligned}
\mathbb{P}\Big[\|\overrightarrow{\boldsymbol{V}}_i - \boldsymbol{H}_{z_i}\|_1 > \varepsilon\|\boldsymbol{H}_{z_i}\|_1\Big] &= \mathbb{P}\Big[\sum_{b=1}^{B}|\boldsymbol{V}_{ib} - H_{z_ib}| > \varepsilon\cdot\Big(\sum_{b=1}^{B}H_{z_ib}\Big)\Big] \\
&\leq \sum_{b=1}^{B}\mathbb{P}\Big[|\boldsymbol{V}_{ib} - H_{z_ib}| > \varepsilon H_{z_ib}\Big] \\
&\leq \sum_{b=1}^{B} 4\exp\Big(-2\ell_b\frac{\varepsilon^2\alpha_{\min}^2}{81}\Big) + 12\exp\Big(-2m\cdot\ell_b\cdot t_b^2(\varepsilon/9)\Big) \\
&\{\text{using equation (A.10)}\} \\
&\leq 4\cdot B\cdot\exp\Big(-2\ell_{\min}\frac{\varepsilon^2\alpha_{\min}^2}{81}\Big) + 12B\cdot\exp\Big(-2m\cdot\ell_{\min}\cdot t_{\min}^2(\varepsilon/9)\Big)
\end{aligned}$$

where $t_{\min}(\varepsilon) \stackrel{\text{def}}{=} \varepsilon\cdot\Big(\frac{-\hat{\alpha}_{\text{pool}}\log(1-\hat{\alpha}_{\text{pool}})}{1-\log(1-\hat{\alpha}_{\text{pool}})}\Big)$ and $\hat{\alpha}_{\text{pool}} = \min_{b\in[B]}\bar{\alpha}_{b,\text{pool}}$. The last inequality follows from the facts that $\ell_b \geq \ell_{\min}$ and $\bar{\alpha}_{b,\text{pool}} \geq \hat{\alpha}_{\text{pool}}$ for all $b \in [B]$. Substituting for $t_{\min}(\varepsilon)$ in the equation above establishes the result.

A.1.6 Proof of Theorem 1.6

We begin by stating analogous versions of Lemmas A.5-A.7.

Lemma A.9. *Let k_1, k_2 be any two segments and $\|\cdot\|$ be an arbitrary norm on $\mathbb{R}^B$. Then for customer i, we have*

$$\frac{\|\overrightarrow{\boldsymbol{V}}_i - \boldsymbol{H}_{k_1}\|}{\|\boldsymbol{H}_{k_1}\|} \leq \frac{\|\boldsymbol{H}_{k_1} - \boldsymbol{H}_{k_2}\|}{2\cdot\max(\|\boldsymbol{H}_{k_1}\|, \|\boldsymbol{H}_{k_2}\|)} \implies \frac{\|\overrightarrow{\boldsymbol{V}}_i - \boldsymbol{H}_{k_1}\|}{\|\boldsymbol{H}_{k_1}\|} \leq \frac{\|\overrightarrow{\boldsymbol{V}}_i - \boldsymbol{H}_{k_2}\|}{\|\boldsymbol{H}_{k_2}\|}$$

Proof. The proof follows from a similar argument as in Lemma A.5. □

Lemma A.10. *For the constant Γ defined in Theorem 1.6, it follows that*

$$\Gamma \leq \min_{k'\neq k}\frac{\|\boldsymbol{H}_k - \boldsymbol{H}_{k'}\|_1}{2\cdot\max\left(\|\boldsymbol{H}_k\|_1, \|\boldsymbol{H}_{k'}\|_1\right)} < 1.$$

Proof. For each $k \neq k'$, define the following:

$$\Gamma_{kk'} \stackrel{\text{def}}{=} \frac{\|\boldsymbol{H}_k - \boldsymbol{H}_{k'}\|_1}{2\cdot\max(\|\boldsymbol{H}_k\|_1, \|\boldsymbol{H}_{k'}\|_1)} = \frac{\sum_{b=1}^{B}\frac{\left|H(\alpha_{kb},\alpha_{b,\text{pool}}) - H(\alpha_{k'b},\alpha_{b,\text{pool}})\right|}{H(\alpha_{b,\text{pool}})}}{2\cdot\max(\|\boldsymbol{H}_k\|_1, \|\boldsymbol{H}_{k'}\|_1)}$$

Since $\boldsymbol{H}_k \neq \boldsymbol{H}_{k'}$ and $\|\boldsymbol{H}_k - \boldsymbol{H}_{k'}\|_1 \leq \|\boldsymbol{H}_k\|_1 + \|\boldsymbol{H}_{k'}\|_1 \leq 2\cdot\max(\|\boldsymbol{H}_k\|_1, \|\boldsymbol{H}_{k'}\|_1)$, it follows that $0 < \Gamma_{kk'} < 1$ for all $k \neq k'$.

Next, observe that $H(\alpha_{kb}, \alpha_{b,\text{pool}}) = -\alpha_{kb}\log\frac{\alpha_{b,\text{pool}}}{1-\alpha_{b,\text{pool}}} - \log(1-\alpha_{b,\text{pool}})$, so that

$$|H(\alpha_{kb}, \alpha_{b,\text{pool}}) - H(\alpha_{k'b}, \alpha_{b,\text{pool}})| = \left|\log\frac{\alpha_{b,\text{pool}}}{1-\alpha_{b,\text{pool}}}\right||\alpha_{kb} - \alpha_{k'b}|$$

Now, note that $H(\alpha_{b,\text{pool}}) \leq 1$ for any category b, using the definition of the binary entropy function. Therefore, it follows that

$$\begin{aligned}\sum_{b=1}^{B} \frac{|H(\alpha_{kb}, \alpha_{b,\text{pool}}) - H(\alpha_{k'b}, \alpha_{b,\text{pool}})|}{H(\alpha_{b,\text{pool}})} &= \sum_{b=1}^{B} \frac{\left|\log \frac{\alpha_{b,\text{pool}}}{1-\alpha_{b,\text{pool}}}\right| |\alpha_{kb} - \alpha_{k'b}|}{H(\alpha_{b,\text{pool}})} \\ &\geq \sum_{b=1}^{B} \left|\log \frac{\alpha_{b,\text{pool}}}{1-\alpha_{b,\text{pool}}}\right| |\alpha_{kb} - \alpha_{k'b}| \\ &\geq \gamma\end{aligned}$$

where γ is as defined in the theorem. Next, consider $\|\boldsymbol{H}_k\|_1$ for some segment k:

$$\|\boldsymbol{H}_k\|_1 = \sum_{b=1}^{B} \frac{H(\alpha_{kb}, \alpha_{b,\text{pool}})}{H(\alpha_{b,\text{pool}})} \leq \frac{1}{H_{\min}} \sum_{b=1}^{B} H(\alpha_{kb}, \alpha_{b,\text{pool}}), \tag{A.11}$$

where $H_{\min} \stackrel{\text{def}}{=} H(\alpha_{\min})$. The inequality above follows because $\alpha_{\min} \leq \alpha_{b,\text{pool}} \leq 1 - \alpha_{\min}$ and the binary entropy function is symmetric around $\frac{1}{2}$ so that $H(\alpha_{\min}) = H(1 - \alpha_{\min})$, from which it follows $H(\alpha_{b,\text{pool}}) \geq H_{\min}$ for any category b. Further since $\alpha_{\min} \leq \alpha_{kb} \leq 1 - \alpha_{\min}$, we can use the argument from Lemma A.6 to obtain $H(\alpha_{kb}, \alpha_{b,\text{pool}}) \leq |\log \alpha_{\min}|$ for all segments k and item categories b. Plugging this into equation (A.11), we get $\|\boldsymbol{H}_k\|_1 \leq \frac{B \cdot |\log \alpha_{\min}|}{H_{\min}}$ for all segments $k \in [K]$. Finally observe that $H_{\min} = -\alpha_{\min} \log \alpha_{\min} - (1 - \alpha_{\min}) \log(1 - \alpha_{\min}) \geq -\log(1 - \alpha_{\min})$, so that $\|\boldsymbol{H}_k\|_1 \leq \frac{B \cdot |\log \alpha_{\min}|}{H_{\min}} \leq \frac{B \cdot |\log \alpha_{\min}|}{|\log(1-\alpha_{\min})|}$.

Combining the above observations, we get that $\Gamma_{kk'} \geq \Gamma$ for all $k \neq k'$. Therefore, $\Gamma \leq \min_{k \neq k'} \Gamma_{kk'} < 1$ and the claim follows. □

Lemma A.11. *Consider a customer i and suppose the following is true:*

$$\frac{\|\overrightarrow{\boldsymbol{V}}_i - \boldsymbol{H}_{z_i}\|_1}{\|\boldsymbol{H}_{z_i}\|_1} \leq \frac{\|\boldsymbol{H}_{z_i} - \boldsymbol{H}_{k'}\|_1}{2 \cdot \max(\|\boldsymbol{H}_{z_i}\|_1, \|\boldsymbol{H}_{k'}\|_1)} \quad \forall\, k' \neq z_i$$

Then it follows that $\hat{\boldsymbol{I}}_2(i) = z_i$, i.e the NN classifier $\hat{\boldsymbol{I}}_2(\cdot)$ correctly classifies customer i. Conversely, it follows that

$$\mathbb{P}\left[\hat{\boldsymbol{I}}_2(i) \neq z_i\right] \leq \mathbb{P}\left[\|\overrightarrow{\boldsymbol{V}}_i - \boldsymbol{H}_{z_i}\|_1 > \Gamma \cdot \|\boldsymbol{H}_{z_i}\|_1\right]$$

Proof. The proof follows from an identical argument as in Lemma A.7, and making use of the results of Lemmas A.9 and A.10 above. □

We are now ready to prove the theorem. The probability that a customer i is misclassified

by the nearest-neighbor classifier $\hat{\boldsymbol{I}}_2(\cdot)$ is given by:

$$\begin{aligned}\mathbb{P}\Big[\hat{\boldsymbol{I}}_2(i) \neq z_i\Big] &\leq \mathbb{P}\Big[\|\overrightarrow{\boldsymbol{V}_i} - \boldsymbol{H}_{z_i}\|_1 > \Gamma \cdot \|\boldsymbol{H}_{z_i}\|_1\Big] \quad \text{(using Lemma A.11)} \\ &\leq 4 \cdot B \cdot \exp\left(-2\ell_{\min}\frac{\Gamma^2\alpha_{\min}^2}{81}\right) + 12B \cdot \exp\left(\frac{-2m \cdot \ell_{\min} \cdot \Gamma^2 \hat{\alpha}_{\text{pool}}^2 \log^2(1-\hat{\alpha}_{\text{pool}})}{81\big(1-\log(1-\hat{\alpha}_{\text{pool}})\big)^2}\right) \\ &\quad \text{(follows from Lemma 1.4)} \qquad \text{(A.12)}\end{aligned}$$

Now given any $0 < \delta < 1$, suppose that the number of observations from each customer satisfy:

$$\ell_{\min} \geq \frac{648B^2}{\gamma^2} \cdot \left(\frac{\log \alpha_{\min}}{\log^2(1-\alpha_{\min}) \cdot \alpha_{\min}}\right)^2 \log(16B/\delta)$$

Then, starting from equation (A.12), it follows that

$$\begin{aligned}&\mathbb{P}\Big[\hat{\boldsymbol{I}}_2(i) \neq z_i\Big] \\ &\leq 4B \cdot \exp\left(-2\ell_{\min}\frac{\Gamma^2\alpha_{\min}^2}{81}\right) + 12B \cdot \exp\left(\frac{-2m \cdot \ell_{\min} \cdot \Gamma^2 \cdot \hat{\alpha}_{\text{pool}}^2 \log^2(1-\hat{\alpha}_{\text{pool}})}{81\big(1-\log(1-\hat{\alpha}_{\text{pool}})\big)^2}\right) \\ &\leq 4B \cdot \exp\left(-2\ell_{\min}\frac{\Gamma^2\alpha_{\min}^2}{81}\right) + 12B \cdot \exp\left(\frac{-2m \cdot \ell_{\min} \cdot \Gamma^2 \cdot \alpha_{\min}^2 \log^2(1-\alpha_{\min})}{81\big(1-\log(1-\alpha_{\min})\big)^2}\right) \\ &\text{(since } \hat{\alpha}_{\text{pool}} \geq \alpha_{\min}\text{)} \\ &\leq 4B \cdot \exp\left(-2\ell_{\min}\frac{\Gamma^2\alpha_{\min}^2 \log^2(1-\alpha_{\min})}{81\,(1-\log(1-\alpha_{\min}))^2}\right) + 12B \cdot \exp\left(\frac{-2m \cdot \ell_{\min} \cdot \Gamma^2 \alpha_{\min}^2 \log^2(1-\alpha_{\min})}{81\big(1-\log(1-\alpha_{\min})\big)^2}\right) \\ &\left(\text{since } \log^2(1-\alpha_{\min}) < (1-\log(1-\alpha_{\min}))^2\right) \\ &\leq 16B \cdot \exp\left(-2\ell_{\min}\frac{\Gamma^2\alpha_{\min}^2 \log^2(1-\alpha_{\min})}{81\,(1-\log(1-\alpha_{\min}))^2}\right) \\ &\text{(since } m \geq 1\text{)} \\ &= 16B \cdot \exp\left(-2\ell_{\min}\frac{\gamma^2 \cdot \log^2(1-\alpha_{\min}) \cdot \alpha_{\min}^2 \log^2(1-\alpha_{\min})}{324B^2 \cdot \log^2 \alpha_{\min} \cdot (1-\log(1-\alpha_{\min}))^2}\right) \\ &\text{(substituting value of } \Gamma\text{)} \\ &\leq 16B \cdot \exp\left(-\ell_{\min}\frac{\gamma^2 \cdot \log^2(1-\alpha_{\min}) \cdot \alpha_{\min}^2 \log^2(1-\alpha_{\min})}{648B^2 \cdot \log^2 \alpha_{\min}}\right) \\ &\left(\text{since } \alpha_{\min} < \frac{1}{2} \implies 1-\log(1-\alpha_{\min}) < 2\right) \\ &\leq \delta \\ &\left(\text{using the bound on } \ell_{\min}\right)\end{aligned}$$

Finally, suppose that $m \cdot \frac{\log^2(1-\hat{\alpha}_{\text{pool}})}{\left(1-\log(1-\hat{\alpha}_{\text{pool}})\right)^2} \geq 1$, and observe that $\hat{\alpha}_{\text{pool}} \geq \alpha_{\min}$. Then, it follows that

$$\begin{aligned}
\mathbb{P}\left[\hat{\boldsymbol{I}}_2(i) \neq z_i\right] &\leq \mathbb{P}\left[\|\overrightarrow{\boldsymbol{V}_i} - \boldsymbol{H}_{z_i}\|_1 > \Gamma \cdot \|\boldsymbol{H}_{z_i}\|_1\right] \quad \text{(using Lemma A.11)} \\
&\leq 4B \cdot \exp\left(-2\ell_{\min}\frac{\Gamma^2\alpha_{\min}^2}{81}\right) + 12B \cdot \exp\left(\frac{-2m \cdot \ell_{\min} \cdot \Gamma^2 \cdot \hat{\alpha}_{\text{pool}}^2 \log^2(1-\hat{\alpha}_{\text{pool}})}{81\left(1-\log(1-\hat{\alpha}_{\text{pool}})\right)^2}\right) \\
&\quad \text{(follows from the result of Lemma 1.4)} \\
&\leq 16 \cdot B \cdot \exp\left(-2\ell_{\min}\frac{\Gamma^2\alpha_{\min}^2}{81}\right) \\
&\quad \text{(using the lower bounds on } m \text{ and } \hat{\alpha}_{\text{pool}})
\end{aligned}$$

Substituting $\ell_{\min} = \log n$ above, we get the desired result.

A.2 Computational study: EM algorithm for the LC method

Let $\boldsymbol{\Theta} = \left[q_1, q_2, \cdots, q_K, \alpha_1, \alpha_2, \cdots, \alpha_K\right]$ denote the set of all parameters (refer to the setup in Section 1.4). The total number of parameters is therefore $K + K = 2 \cdot K$. Let $\mathcal{D} = \{\boldsymbol{x}_1^{\text{obs}}, \boldsymbol{x}_2^{\text{obs}}, \ldots, \boldsymbol{x}_m^{\text{obs}}\}$ be the observed rating vectors from the m customers. Then assuming that the rating vectors are sampled i.i.d. from the population mixture distribution, the log-likelihood of the data can be written as:

$$\log \mathbb{P}[\mathcal{D} \mid \boldsymbol{\Theta}] = \sum_{i=1}^{m} \log \sum_{k=1}^{K} q_k \cdot \left(\prod_{j \in N(i)} \alpha_k^{\mathbf{1}[x_{ij}=+1]} \cdot (1-\alpha_k)^{\mathbf{1}[x_{ij}=-1]}\right), \tag{A.13}$$

where $\mathbf{1}[\cdot]$ denotes the indicator function. The LC method computes the estimates of the parameters $\boldsymbol{\Theta}$ that maximize the log-likelihood (A.13).

The MLE can be computed via the EM algorithm by introducing latent variables corresponding to the true segment of each customer, which we denote by $\boldsymbol{z} = [z_1, z_2, \ldots, z_m]$ where $z_i \in [K]$ denotes the true segment of customer i. The complete data log-likelihood can then be written as

$$\log \mathbb{P}[\mathcal{D}, \boldsymbol{z} \mid \boldsymbol{\Theta}] = \sum_{i=1}^{m} \sum_{k=1}^{K} \mathbf{1}[z_i = k] \cdot \log\left(q_k \cdot \prod_{j \in N(i)} \alpha_k^{\mathbf{1}[x_{ij}=+1]} \cdot (1-\alpha_k)^{\mathbf{1}[x_{ij}=-1]}\right).$$

Starting from an initial estimate $\boldsymbol{\Theta}^{(0)}$ of the parameters, the EM algorithm executes the following two steps in each iteration $t \geq 1$:

- **E-step**: Given the data $\mathcal{D}$ and the current estimate of the parameters $\boldsymbol{\Theta}^{(t-1)}$, we compute the expected complete data log-likelihood w.r.t to the conditional distribution $\boldsymbol{z} \mid \mathcal{D}; \boldsymbol{\Theta}^{(t-1)}$:

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{z} \mid \mathcal{D}; \boldsymbol{\Theta}^{(t-1)}} \{\log \mathbb{P}[\mathcal{D}, \boldsymbol{z} \mid \boldsymbol{\Theta}]\} \\
&= \sum_{i=1}^{m} \sum_{k=1}^{K} \gamma_{ik}^{(t-1)} \left[\log q_k + \sum_{j \in N(i)} \mathbf{1}[x_{ij} = +1] \log \alpha_k + \mathbf{1}[x_{ij} = -1] \log(1 - \alpha_k) \right]
\end{aligned}$$

  Here $\gamma_{ik}^{(t-1)} = \mathbb{P}[z_i = k \mid \mathcal{D}, \boldsymbol{\Theta}^{(t-1)}]$ is the posterior probability of customer i's latent segment being equal to $k \in [K]$, conditioned on the observed ratings and the current model parameters. We can compute $\gamma_{ik}^{(t-1)}$ using Bayes theorem as follows

$$\begin{aligned}
\gamma_{ik}^{(t-1)} &\propto \mathbb{P}[\boldsymbol{x}_i^{\text{obs}} \mid z_i = k; \boldsymbol{\Theta}^{(t-1)}] \cdot \mathbb{P}[z_i = k \mid \boldsymbol{\Theta}^{(t-1)}] \\
&= \frac{\prod_{j \in N(i)} (\alpha_k^{(t-1)})^{\mathbf{1}[x_{ij}=+1]} \cdot (1 - \alpha_k^{(t-1)})^{\mathbf{1}[x_{ij}=-1]} \cdot q_k^{(t-1)}}{\sum_{k'=1}^{K} \prod_{j \in N(i)} (\alpha_{k'}^{(t-1)})^{\mathbf{1}[x_{ij}=+1]} \cdot (1 - \alpha_{k'}^{(t-1)})^{\mathbf{1}[x_{ij}=-1]} \cdot q_{k'}^{(t-1)}}.
\end{aligned}$$

- **M-step**: Based on the current posterior estimates of the customer segment memberships $\gamma_{ik}^{(t-1)}$ and the observed data $\mathcal{D}$, the model parameters are updated by maximizing the expected complete data log-likelihood $\mathbb{E}_{\boldsymbol{z} \mid \mathcal{D}; \boldsymbol{\Theta}^{(t-1)}} \{\log \mathbb{P}[\mathcal{D}, \boldsymbol{z} \mid \boldsymbol{\Theta}]\}$, which can be shown to be a lower bound on the (incomplete) data log-likelihood $\log \mathbb{P}[\mathcal{D} \mid \boldsymbol{\Theta}]$. Taking into account the constraint $\sum_{k'=1}^{K} q_{k'} = 1$, we get the following update for the parameter q_k

$$q_k^{(t)} = \frac{\sum_{i=1}^{m} \gamma_{ik}^{(t-1)}}{m} \qquad \forall k \in [K]$$

  Similarly, for the parameter α_k, we get the following update

$$\alpha_k^{(t)} = \frac{\sum_{i=1}^{m} \sum_{j \in N(i)} \gamma_{ik}^{(t-1)} \cdot \mathbf{1}[x_{ij} = +1]}{\sum_{i=1}^{m} \sum_{j \in N(i)} \gamma_{ik}^{(t-1)}} \qquad \forall k \in [K]$$

We repeat these two steps until convergence of the log-likelihood $\log \mathbb{P}[\mathcal{D} \mid \boldsymbol{\Theta}]$.

Implementation specifics. We imposed a Beta$(2, 2)$ prior on the parameters α_k, and Dir$(1.5, 1.5, \ldots, 1.5)$ prior on the proportions q_k, to avoid numerical and overfitting issues for sparse graphs. In this case, we compute the maximum a posteriori probability (MAP) estimate of the model parameters $\boldsymbol{\Theta}$, using an EM algorithm similar to the one described above. Further, since the log-likelihood objective in (A.13) is non-convex, the LC method is sensitive to the initial estimates $\boldsymbol{\Theta}^{(0)}$ and consequently, we run both the EM and SLSQP approaches 10 times with different random initializations and report the best outcome.

A.3 MovieLens case study

A.3.1 Benchmark details.

LC method. The LC method assumes that the population is comprised of K segments with proportion q_k and probability of like α_k for segment $k \in [K]$. Then, it estimates the parameters by maximizing the log-likelihood of the observed ratings:

$$\max_{\substack{q_1,q_2,\ldots,q_K \\ \alpha_1,\ldots,\alpha_K}} \sum_{i=1}^{m} \log \left(\sum_{k=1}^{K} q_k \prod_{j \in N^{\text{train}}(i)} \alpha_k^{\mathbf{1}[r_{ij}=+1]} (1-\alpha_k)^{\mathbf{1}[r_{ij}=-1]} \right)$$
$$\text{s.t.} \sum_{k \in [K]} q_k = 1, q_k \geq 0, 0 \leq \alpha_k \leq 1 \ \forall \ k$$

We use the EM algorithm described in Appendix A.2 to estimate the parameters. Let α_k^{LC}, $k = 1, 2, \ldots, K$, denote the segment parameters estimated by the LC method and let γ_{ik} denote the posterior probability of membership in segment k for user i. Then, the predicted rating for a new movie j_{new} is given by: $\hat{r}_{ij_{\text{new}}}^{\text{LC}} = +1$ if $\sum_{k=1}^{K} \gamma_{ik} \alpha_k^{\text{LC}} \geq 0.5$ else $\hat{r}_{ij_{\text{new}}}^{\text{LC}} = -1$.

EB method. The EB method assumes that the population is described by a prior distribution $G_{prior}(\cdot)$ over the parameter $0 \leq \alpha \leq 1$ where α represents the probability of liking any item; and each individual i samples $\alpha_i \sim G_{prior}$ and uses α_i to generate ratings for the movies. Given the observed ratings, the parameters of $G_{prior}(\cdot)$ are estimated using a standard technique like maximum likelihood or method-of-moments. Then, for each individual i, we compute the posterior mean $\hat{\alpha}_i$ based on the estimated prior $\hat{G}_{prior}$, i.e.

$$\hat{\alpha}_i = \int_0^1 \alpha \cdot \hat{G}_{post,i}(\alpha) \, d\alpha,$$

where $\hat{G}_{post,i}$ is the posterior distribution for α_i. Since $0 \leq \alpha \leq 1$ and the ratings $r_{ij} \in \{+1, -1\}$ are binary, we assume the prior is a beta distribution $\text{Beta}(a, b)$ with $a, b > 0$ and estimate the parameters a, b using the method-of-moments. Then, the posterior distribution for α_i is given by (since the beta distribution is a conjugate prior for the binomial distribution):

$$\hat{G}_{post,i} = \text{Beta} \left(a + \sum_{j \in N^{\text{train}}(i)} \mathbf{1}[r_{ij} = +1], \ b + \sum_{j \in N^{\text{train}}(i)} \mathbf{1}[r_{ij} = -1] \right)$$

where r_{ij} is the rating given by user i for movie j and $N^{\text{train}}(i)$ is the set of movies rated by user i. Consequently, we have that

$$\hat{\alpha}_i = \frac{a + \sum_{j \in N^{\text{train}}(i)} \mathbf{1}[r_{ij} = +1]}{a + b + |N^{\text{train}}(i)|}$$

and given a new movie j_{new}, we predict $\hat{r}_{ij_{\text{new}}}^{\text{EB}} = +1$ if $\hat{\alpha}_i \geq 0.5$ otherwise $\hat{r}_{ij_{\text{new}}}^{\text{EB}} = -1$.

Genre	Accuracy			% Improvement	
	k-medoids	SC	α-EMBED	over k-medoids	over SC
Action ($K = 2$)	30.7	44.9	56.4	83.7	25.6
Comedy ($K = 4$)	37.7	45.0	58.4	54.9	29.8
Drama ($K = 4$)	38.6	47.4	57.2	48.2	20.7

Table A.1: Comparison against additional benchmarks for new movie recommendations on MovieLens dataset

Similarity-based clustering methods. We also consider two additional similarity-based clustering benchmarks: (1) k-medoids clustering [dHINM04], an extension of the popular k-means algorithm to handle missing entries, and (2) spectral clustering [NJW01] which clusters data points via spectral decomposition of an affinity (or similarity) matrix. Both of these methods take as input an appropriate similarity (or distance) measure between any two data points; we used the standard *cosine distance* measure: for any two customers $i \neq i'$ with rating vectors $\boldsymbol{r}_i$ and $\boldsymbol{r}_{i'}$ (with $r_{ij} = 0$ if customer i did not rate item j), the cosine similarity is defined as:

$$\text{cossim}(i, i') \stackrel{\text{def}}{=} \frac{\sum_{j=1}^{n} r_{ij} \cdot r_{i'j}}{\sqrt{\sum_{j=1}^{n} r_{ij}^2} \cdot \sqrt{\sum_{j=1}^{n} r_{i'j}^2}}$$

The measure is termed cosine similarity because it can be viewed as the cosine of the angle between the vectors $\boldsymbol{r}_i$ and $\boldsymbol{r}_{i'}$. Observing the term $\sum_{j=1}^{n} x_{ij} \cdot x_{i'j}$ in the numerator above, we note that it is the difference between the number of *agreements* (i.e. both rate $+1$ or -1) and *disagreements* (i.e. one rates $+1$ and other rates -1) between i and i' for commonly rated items. This is scaled by (square root of) the product of the number of items rated by each customer in the denominator. Therefore, more the number of agreements, larger is the similarity between the customers. The cosine similarity lies between -1 and 1, a cosine similarity of 1 indicates perfect agreement and -1 indicates perfect disagreement. The cosine distance is given by $\text{cosd}(i, i') := 1 - \text{cossim}(i, i')$ which lies between 0 and 2.

For both the k-medoids and spectral clustering (which we refer to as SC) methods, after clustering the customer population, we estimate a separate parameter α_k for each segment k and predict ratings for new movies using the estimated parameters (as done for our approach α-EMBED). The accuracies for the similarity-based clustering benchmarks are reported in Table A.1. We see that they perform poorly, this is expected—these techniques rely on a well-defined similarity measure between data points which becomes difficult to determine when there are (many) missing entries. In particular, our method achieves upto 84% improvement over k-medoids (for action genre) and 30% improvement over spectral clustering (for comedy genre).

A.3.2 Partially specified model for ratings

We discuss here the application of our segmentation approach when the model generating user ratings is only partially specified. We focus on movies in the three genres, action, comedy and drama. Since movies were tagged with more than one genre in the dataset, we only considered movies that had exactly one of these 3 genres; this left us with a total of 1,861 movies and a population of 6,040 users rating on these movies. We consider the generative model introduced in Definition 1.2, here we have $B = 3$ categories and the parameter vector $\boldsymbol{\alpha}_k = (\alpha_{k,\text{action}}, \alpha_{k,\text{comedy}}, \alpha_{k,\text{drama}})$. For each genre, we separately estimated the pooled parameters $\alpha_{b,\text{pool}}$ as described in the main text, where $b \in \{\text{action}, \text{comedy}, \text{drama}\}$. Then, we apply Algorithm 4 to compute a 3-dimensional embedding vector for each user. The $\mathcal{V}$ matrix was incomplete, i.e., there were certain users that did not rate for movies in all 3 genres and therefore we computed a rank $r = 2$ factorization[1] of the $\mathcal{V}$ matrix. Then, we clustered the embedding vectors using the k-means algorithm into $K = 5$ segments, where the choice of K was tuned using a validation set similar to the case for individual genres discussed in the main text.

We compare our approach to a LC benchmark, and use an EM algorithm (similar to the one derived above in Appendix A.2) for estimating the model parameters. Table A.2 reports the accuracies as well as statistics of the training and test datasets. As expected, segmenting the population provides significant improvements compared to the population model, which assumes that users have homogeneous preferences. Our approach still outperforms the LC benchmark, by upto 20% for the action genre and by 6.8% in aggregate. Further, even with the matrix factorization step, our approach is still $\sim 3\times$ faster as compared to EM.

[1]We used the ALS (Alternating Least Squares) class in `pyspark.mllib.recommendation` module, which is part of Apache Spark's python API, to compute the factorization: https://spark.apache.org/docs/latest/api/python/pyspark.mllib.html#pyspark.mllib.recommendation.ALS.

Performance under partially specified model for user ratings

Genre	Train Data			Test Data		Accuracy			% Improvement	
	Users	Movies	Ratings	Movies	Ratings	POP	LC	α-EMBED	over POP	over LC
Action	5857	314	179K	29	352	32.6	47.9	57.5	76.4	20.0
Comedy	5972	723	256K	170	1931	38.2	53.3	57.1	49.5	7.1
Drama	5993	824	238K	367	4070	38.8	52.9	55.3	42.5	4.5
Aggregate	6040	1861	673K	566	6353	38.0	52.6	56.2	47.9	6.8

Table A.2: Training/test data statistics and rating prediction accuracies of the different methods when choosing $K = 5$ segments.

Appendix B

Chapter 2 Proofs and Details of Numerical Experiments

B.1 Analysis of expected penalties under soft-penalty algorithm

First, we introduce some additional notation. Let $\mathbb{P}_{ih}[\mathcal{A}]$ (resp. $\mathbb{P}_{ia}[\mathcal{A}]$) denote the probability of some event $\mathcal{A}$ conditioned on the fact that worker w_i is honest (resp. adversarial). Similarly, $\mathbb{E}_{ih}[\boldsymbol{X}]$ (resp. $\mathbb{E}_{ia}[\boldsymbol{X}]$) denotes the conditional expectation of the random variable $\boldsymbol{X}$ given the event that worker w_i is honest (resp. adversarial). Also, let $f(\cdot)$ denote the probability density function (PDF) of the worker reliability distribution. Recall the definitions $P := q\mu + (1-q)$, $Q := 1 - q\mu$, and the function $g(x) := \frac{1-x^r}{r(1-x)}$; observe that $g(\cdot)$ is *strictly increasing* on $(0,1)$. Let $\mathbf{1}[\mathcal{A}]$ denote the indicator variable taking value 1 if the event $\mathcal{A}$ is true and 0 otherwise. Let $\boldsymbol{D}_j^+$ (resp. $\boldsymbol{D}_j^-$) denote the number of workers who label task t_j as $+1$ (resp -1). In other words, $\boldsymbol{D}_j^+ = \sum_{w_i \in W_j} \mathbf{1}[\boldsymbol{W}_{ij} = +1]$ and $\boldsymbol{D}_j^- = \sum_{w_i \in W_j} \mathbf{1}[\boldsymbol{W}_{ij} = -1]$, where recall that W_j denotes the set of workers who labeled task t_j. Here, $\boldsymbol{W}_{ij}$ represents the label assigned by worker w_i to task t_j, note that it is a random variable under the generative model described in the main text. Finally, let $\mathrm{Bin}(z,p)$ denote the Binomial distribution with parameters z and p.

We begin by proving some important lemmas that will be repeatedly used in the proofs below.

Lemma B.1. *Under the generative model in Section 2.4.1, the probability that worker w_i provides response $+1$ for a task t_j is given by*

$$\mathbb{P}[\boldsymbol{W}_{ij} = +1 \,|\, \boldsymbol{Y}_j = +1] = P$$
$$\mathbb{P}[\boldsymbol{W}_{ij} = +1 \,|\, \boldsymbol{Y}_j = -1] = Q$$

Furthermore, conditioned on $\boldsymbol{Y}_j = +1$ or $\boldsymbol{Y}_j = -1$, the random variables $\mathbf{1}[\boldsymbol{W}_{ij} = +1]$ are i.i.d. for all $w_i \in W_j$. As a result, it follows that $\boldsymbol{D}_j^+ \,|\, \boldsymbol{Y}_j = +1 \sim \mathrm{Bin}(r, P)$ and $\boldsymbol{D}_j^+ \,|\, \boldsymbol{Y}_j = -1 \sim \mathrm{Bin}(r, Q)$.

Proof. Consider the case when $\boldsymbol{Y}_j = +1$:

$$\begin{aligned}
&\mathbb{P}[\boldsymbol{W}_{ij} = +1 \,|\, \boldsymbol{Y}_j = +1] \\
&= \mathbb{P}_{ih}[\boldsymbol{W}_{ij} = +1 \,|\, \boldsymbol{Y}_j = +1] \cdot \mathbb{P}[w_i \text{ is honest} \,|\, \boldsymbol{Y}_j = +1] \\
&+ \mathbb{P}_{ia}[\boldsymbol{W}_{ij} = +1 \,|\, \boldsymbol{Y}_j = +1] \cdot \mathbb{P}[w_i \text{ is adversarial } \,|\, \boldsymbol{Y}_j = +1] \\
&= \left(\int_0^1 \mathbb{P}_{ih}[\boldsymbol{W}_{ij} = +1 \,|\, \boldsymbol{M}_i = \mu_i; \boldsymbol{Y}_j = +1] \cdot f(\mu_i) d\mu_i \right) \cdot q + 1 \cdot (1 - q) \\
&= \left(\int_0^1 \mu_i f(\mu_i) d\mu_i \right) \cdot q + (1 - q) = q\mu + (1 - q) = P,
\end{aligned}$$

where the first term of the second equality follows from the law of total expectation, the second term because the adversary always labels a task $+1$, and the third equality follows from the definition of honest worker reliability $\boldsymbol{M}_i$.

Furthermore, it follows from our generative process that conditioned on $\boldsymbol{Y}_j = +1$, the labels from any two workers $w_i \neq w_{i'}$ for task t_j are generated independently. Therefore, we have shown that, conditioned on $\boldsymbol{Y}_j = +1$, the random variables $\mathbf{1}[\boldsymbol{W}_{ij} = +1]$ are independent and identically distributed with probability P of taking value 1. Because $\boldsymbol{D}_j^+ = \sum_{w_i \in W_j} \mathbf{1}[\boldsymbol{W}_{ij} = +1]$ and $|W_j| = r$, it follows that $\boldsymbol{D}_j^+ \,|\, \boldsymbol{Y}_j = +1$ is a sum of r i.i.d. Bernoulli random variables and, hence, $\boldsymbol{D}_j^+ \,|\, \boldsymbol{Y}_j = +1 \sim \text{Bin}(r, P)$.

A similar argument shows that $\mathbb{P}[\boldsymbol{W}_{ij} = +1 \,|\, \boldsymbol{Y}_j = -1] = Q$ and consequently $\boldsymbol{D}_j^+ \,|\, \boldsymbol{Y}_j = -1 \sim \text{Bin}(r, Q)$. □

Lemma B.2. *Under the generative model in Section 2.4.1, suppose that worker w_i is honest with a sampled reliability μ_i and let $\boldsymbol{S}_{ij}$ denote the penalty received by w_i from task $t_j \in \mathcal{T}_i$ in the soft-penalty algorithm. Then, we can show*

$$\mathbb{E}_{ih}[\boldsymbol{S}_{ij} \,|\, \boldsymbol{M}_i = \mu_i] = \gamma \cdot \Big(\mu_i \cdot g(1-P) + (1-\mu_i) \cdot g(P) \Big) + (1-\gamma) \cdot \Big(\mu_i \cdot g(Q) + (1-\mu_i) \cdot g(1-Q) \Big)$$

Similarly, if worker w_i is adversarial, then it follows:

$$\mathbb{E}_{ia}[\boldsymbol{S}_{ij}] = \gamma \cdot g(1 - P) + (1 - \gamma) \cdot g(1 - Q)$$

Proof. We begin with the case when w_i is honest. Using the law of total expectation, we have:

$$\begin{aligned}
&\mathbb{E}_{ih}[\boldsymbol{S}_{ij} \,|\, \boldsymbol{M}_i = \mu_i] \\
&= \sum_{v_1, v_2 \in \{-1, +1\}} \mathbb{E}_{ih}[\boldsymbol{S}_{ij} \,|\, \boldsymbol{M}_i = \mu_i; \boldsymbol{W}_{ij} = v_1; \boldsymbol{Y}_j = v_2] \cdot \mathbb{P}_{ih}[\boldsymbol{W}_{ij} = v_1; \boldsymbol{Y}_j = v_2 \,|\, \boldsymbol{M}_i = \mu_i]
\end{aligned}$$

We first consider the case when $v_1 = +1$ and $v_2 = +1$. In this case, because worker w_i assigns label $+1$ to task t_j (i.e., $\boldsymbol{W}_{ij} = +1$), the penalty $\boldsymbol{S}_{ij}$ assigned by the task to the worker is equal to $1/\boldsymbol{D}_j^+$. Furthermore, because $\boldsymbol{Y}_j = +1$ and $\mathbf{1}[\boldsymbol{W}_{ij} = +1] = 1$, it follows

from the arguments in Lemma B.1 that $\boldsymbol{D}_j^+ - 1$ is distributed as $\text{Bin}(r-1, P)$. Focusing on the first term in the product, it follows

$$\begin{aligned}
\mathbb{E}_{ih}[\boldsymbol{S}_{ij} \,|\, \boldsymbol{M}_i = \mu_i; \boldsymbol{W}_{ij} = +1; \boldsymbol{Y}_j = +1] &= \mathbb{E}_{ih}\left[1/\boldsymbol{D}_j^+ \,|\, \boldsymbol{M}_i = \mu_i; \boldsymbol{W}_{ij} = +1; \boldsymbol{Y}_j = +1\right] \\
&= \sum_{k=0}^{r-1} \frac{1}{1+k} \binom{r-1}{k} \cdot P^k (1-P)^{r-1-k} \\
&= \frac{1}{r} \cdot \sum_{k=0}^{r-1} \binom{r}{k+1} \cdot P^k (1-P)^{r-1-k} \\
&= \frac{1}{rP} \cdot \sum_{k'=1}^{r} \binom{r}{k'} \cdot P^{k'} (1-P)^{r-k'} \\
&= \frac{1-(1-P)^r}{rP} = g(1-P), \qquad \text{(B.1)}
\end{aligned}$$

where the third equality follows because $\frac{1}{k+1} \cdot \binom{r-1}{k} = \frac{1}{r} \cdot \binom{r}{k+1}$, the fifth equality follows because $\sum_{k'=0}^{r} \binom{r}{k'} \cdot P^{k'} (1-P)^{r-k'} = 1$, and the last equality follows from the definition of the function $g(\cdot)$.

Furthermore, for the second term:

$$\begin{aligned}
&\mathbb{P}_{ih}[\boldsymbol{W}_{ij} = +1, \boldsymbol{Y}_j = +1 \,|\, \boldsymbol{M}_i = \mu_i] \\
&= \mathbb{P}_{ih}[\boldsymbol{W}_{ij} = +1 \,|\, \boldsymbol{Y}_j = +1; \boldsymbol{M}_i = \mu_i] \cdot \mathbb{P}_{ih}[\boldsymbol{Y}_j = +1 \,|\, \boldsymbol{M}_i = \mu_i] = \mu_i \gamma \qquad \text{(B.2)}
\end{aligned}$$

Combining equations (B.1) and (B.2), it follows that

$$\mathbb{E}_{ih}[\boldsymbol{S}_{ij} \,|\, \boldsymbol{M}_i = \mu_i; \boldsymbol{W}_{ij} = +1; \boldsymbol{Y}_j = +1] \cdot \mathbb{P}_{ih}[\boldsymbol{W}_{ij} = +1; \boldsymbol{Y}_j = +1 \,|\, \boldsymbol{M}_i = \mu_i] = g(1-P)\mu_i\gamma \qquad \text{(B.3)}$$

The case when $v_1 = -1$ and $v_2 = +1$ (i.e., $\boldsymbol{W}_{ij} = -1$ and $\boldsymbol{Y}_j = +1$) follows a symmetric argument. In particular, because worker w_i assigns the label -1 to task t_j, the penalty $\boldsymbol{S}_{ij}$ that is assigned is equal to $1/\boldsymbol{D}_j^-$. Furthermore, it follows from the arguments in Lemma B.1 that $\boldsymbol{D}_j^- - 1$ is distributed as $\text{Bin}(r-1, 1-P)$. Then, repeating the sequence of steps above, we can show that

$$\mathbb{E}_{ih}[\boldsymbol{S}_{ij} \,|\, \boldsymbol{M}_i = \mu_i; \boldsymbol{W}_{ij} = -1; \boldsymbol{Y}_j = +1] \cdot \mathbb{P}_{ih}[\boldsymbol{W}_{ij} = -1; \boldsymbol{Y}_j = +1 \,|\, \boldsymbol{M}_i = \mu_i] = g(P) \cdot (1-\mu_i)\gamma \qquad \text{(B.4)}$$

Replacing P by Q, μ_i by $1-\mu_i$, and γ by $1-\gamma$ in equations (B.3),(B.4) yields the expressions for the other two cases. In particular, we have

$$\begin{aligned}
&\mathbb{E}_{ih}[\boldsymbol{S}_{ij} \,|\, \boldsymbol{M}_i = \mu_i; \boldsymbol{W}_{ij} = +1; \boldsymbol{Y}_j = -1] \cdot \mathbb{P}_{ih}[\boldsymbol{Y}_j = -1; \boldsymbol{W}_{ij} = +1 \,|\, \boldsymbol{M}_i = \mu_i] \\
&= g(1-Q) \cdot (1-\gamma) \cdot (1-\mu_i) \\
&\mathbb{E}_{ih}[\boldsymbol{S}_{ij} \,|\, \boldsymbol{M}_i = \mu_i; \boldsymbol{W}_{ij} = -1; \boldsymbol{Y}_j = -1] \cdot \mathbb{P}_{ih}[\boldsymbol{Y}_j = -1; \boldsymbol{W}_{ij} = -1 \,|\, \boldsymbol{M}_i = \mu_i] \\
&= g(Q) \cdot (1-\gamma) \cdot \mu_i \qquad \text{(B.5)}
\end{aligned}$$

Combining equations B.3-B.5, we obtain

$$\mathbb{E}_{ih}[\boldsymbol{S}_{ij} \mid \boldsymbol{M}_i = \mu_i] = \gamma \cdot \Big(\mu_i \cdot g(1-P) + (1-\mu_i) \cdot g(P)\Big) + (1-\gamma) \cdot \Big(\mu_i \cdot g(Q) + (1-\mu_i) \cdot g(1-Q)\Big)$$

and the first half of the claim follows.

Next, suppose that worker w_i is an adversary. Because adversaries always assign the label $+1$, we need to consider only two cases: $\boldsymbol{Y}_j = +1$ and $\boldsymbol{Y}_j = -1$. The conditional expectations of the penalties in both cases are identical to those above:

$$\begin{aligned} \mathbb{E}_{ia}[\boldsymbol{S}_{ij} \mid \boldsymbol{W}_{ij} = +1; \boldsymbol{Y}_j = +1] &= g(1-P) \\ \mathbb{E}_{ia}[\boldsymbol{S}_{ij} \mid \boldsymbol{W}_{ij} = +1; \boldsymbol{Y}_j = -1] &= g(1-Q) \end{aligned} \tag{B.6}$$

Further, we also have

$$\begin{aligned} \mathbb{P}_{ia}[\boldsymbol{W}_{ij} = +1; \boldsymbol{Y}_j = +1] &= \gamma \\ \mathbb{P}_{ia}[\boldsymbol{W}_{ij} = +1; \boldsymbol{Y}_j = -1] &= 1-\gamma \end{aligned} \tag{B.7}$$

Combining equations B.6 and B.7, we obtain

$$\mathbb{E}_{ia}[\boldsymbol{S}_{ij}] = \gamma \cdot g(1-P) + (1-\gamma) \cdot g(1-Q),$$

and the result of the lemma follows. □

We are now ready to prove the theorems.

B.1.1 Proof of Theorem 2.2

First, note that if $q = 1$ then $P = \mu$ and $Q = 1 - \mu$. Also, since all workers are honest, we remove the explicit conditioning on worker w_i being honest. Then the expected penalty allocated to worker w_i having reliability μ_i:

$$\begin{aligned} &\mathbb{E}[\mathbf{PEN}_i \mid \boldsymbol{M}_i = \mu_i] = \frac{1}{l} \sum_{j \in \mathcal{T}_i} \mathbb{E}[\boldsymbol{S}_{ij} \mid \boldsymbol{M}_i = \mu_i] \\ &= \gamma \cdot \Big(\mu_i \cdot g(1-P) + (1-\mu_i) \cdot g(P)\Big) + (1-\gamma) \cdot \Big(\mu_i \cdot g(Q) + (1-\mu_i) \cdot g(1-Q)\Big) \\ &\text{(using Lemma B.2)} \\ &= \gamma \cdot \Big(\mu_i \cdot g(1-\mu) + (1-\mu_i) \cdot g(\mu)\Big) + (1-\gamma) \cdot \Big(\mu_i \cdot g(1-\mu) + (1-\mu_i) \cdot g(\mu)\Big) \\ &= g(\mu) - \mu_i \cdot (g(\mu) - g(1-\mu)) \end{aligned}$$

Since $g(\cdot)$ is strictly increasing on $(0, 1)$ and $\mu > \frac{1}{2}$, we have that $\mu > 1 - \mu$ and consequently $g(\mu) - g(1-\mu) > 0$. The claim then follows.

B.1.2 Proof of Theorem 2.3

For an honest worker w_i, we have that

$$
\begin{aligned}
&p_h := \mathbb{E}_{ih}[\mathbf{PEN}_i] \\
&= \frac{1}{l}\sum_{j\in\mathcal{T}_i} \mathbb{E}_{ih}[\boldsymbol{S}_{ij}] \\
&= \frac{1}{l}\sum_{j\in\mathcal{T}_i} \int_0^1 \mathbb{E}_{ih}[\boldsymbol{S}_{ij} \mid \boldsymbol{M}_i = \mu_i] f(\mu_i) d\mu_i \quad \text{(law of total expectation)} \\
&= \int_0^1 \Big(\gamma\cdot(\mu_i\cdot g(1-P) + (1-\mu_i)\cdot g(P)) + (1-\gamma)\cdot(\mu_i\cdot g(Q) + (1-\mu_i)\cdot g(1-Q))\Big) f(\mu_i) d\mu_i \\
&\text{(using Lemma B.2)} \\
&= \gamma\cdot\Big(\mu\cdot g(1-P) + (1-\mu)\cdot g(P)\Big) + (1-\gamma)\cdot\Big(\mu\cdot g(Q) + (1-\mu)\cdot g(1-Q)\Big) \\
&\text{(since } \int_0^1 \mu_i f(\mu_i) d\mu_i = \mu\text{)}
\end{aligned}
$$

Similarly, when worker w_i is adversarial,

$$
\begin{aligned}
p_a &:= \mathbb{E}_{ia}[\mathbf{PEN}_i] \\
&= \frac{1}{l}\sum_{j\in\mathcal{T}_i} \mathbb{E}_{ia}[\boldsymbol{S}_{ij}] \\
&= \frac{1}{l}\sum_{j\in\mathcal{T}_i} \gamma\cdot g(1-P) + (1-\gamma)\cdot g(1-Q) \quad \text{(using Lemma B.2)} \\
&= \gamma\cdot g(1-P) + (1-\gamma)\cdot g(1-Q)
\end{aligned}
$$

Suppose $q\mu \le 1/2$, then we have that $Q = 1 - q\mu \ge 1/2$ and $P > Q \ge 1/2$. Further, because $g(\cdot)$ is strictly increasing, it follows that $g(1-P) - g(P) < 0$ and $g(1-Q) - g(Q) \le 0$. Given this, and using the expressions for the expected penalty computed above, we have that:

$$
\begin{aligned}
&p_a - p_h \\
&= \gamma\cdot(1-\mu)\cdot \underbrace{(g(1-P) - g(P))}_{<0} + (1-\gamma)\cdot\mu\cdot \underbrace{(g(1-Q) - g(Q))}_{\le 0} \\
&\le 0
\end{aligned}
$$

Therefore, $q\mu > \frac{1}{2}$ is a *necessary* condition for $p_h < p_a$. Assuming this condition is met, we

derive the second condition:

$$\begin{aligned}
p_a > p_h &\iff p_a - p_h > 0 \\
&\iff \gamma \cdot (1-\mu) \cdot \Big(g(1-P) - g(P)\Big) + (1-\gamma) \cdot \mu \cdot \Big(g(1-Q) - g(Q)\Big) > 0 \\
&\iff (1-\gamma) \cdot \mu \cdot \Big(g(1-Q) - g(Q)\Big) > \gamma \cdot (1-\mu) \cdot \Big(g(P) - g(1-P)\Big) \\
&\iff \frac{\mu}{1-\mu} \cdot \frac{\Big(g(1-Q) - g(Q)\Big)}{\Big(g(P) - g(1-P)\Big)} > \frac{\gamma}{1-\gamma} \\
&\text{(since } P > q\mu > \frac{1}{2} \text{ and } Q = 1 - q\mu < \frac{1}{2}\text{)}
\end{aligned}$$

Consider the function $h(\mu, q) = \frac{g(1-Q)-g(Q)}{g(P)-g(1-P)}$ in the regime $q\mu > \frac{1}{2}$. Note that as q increases, Q decreases (since $\frac{\partial Q}{\partial q} = -\mu < 0$) and therefore $g(1-Q) - g(Q)$ (strictly) increases. Similarly, P decreases as q increases (since $\frac{\partial P}{\partial q} = \mu - 1 \leq 0$) and therefore $g(P) - g(1-P)$ decreases. It follows that $h(\mu, q)$ is a *strictly* increasing function of q. The result of the theorem now follows.

Proof of Corollary 2.3.1. Since $q\mu > \frac{1}{2}$, we have that $P > \frac{1}{2}$ and $Q < \frac{1}{2}$. From the proof of Theorem 2.3 above, we have

$$\begin{aligned}
p_a > p_h &\iff \frac{\mu}{1-\mu} \cdot h(\mu, q) > \frac{\gamma}{1-\gamma} \\
&\iff h_\mu(q) > \frac{\gamma}{1-\gamma} \\
&\iff q > h_\mu^{-1}\left(\frac{\gamma}{1-\gamma}\right) \\
&\text{(since } h_\mu(q) \text{ is strictly increasing)}
\end{aligned}$$

B.2 Asymptotic identification of honest and adversarial workers

We begin with the proof of the lemma establishing the locally-tree like property of the worker-task assignment graph.

B.2.1 Proof of Lemma 2.5

We adapt the proof from [KOS14]. Consider the following (discrete time) random process that generates the random graph $\mathcal{B}_{\boldsymbol{w},2}$ starting from the root node $\boldsymbol{w}$. In the first step, we

connect l task nodes to node $\boldsymbol{w}$ according to the configuration model, where l half-edges are matched to a randomly chosen subset of mr task half-edges of size l. Let α_1 denote the probability that the resulting graph is *not* a tree, that is, at least one pair of edges are connected to the same task node. Since there are $\binom{l}{2}$ pairs and each pair of half-edges is connected to the same task node with probability $\frac{r-1}{mr-1}$, we have that:

$$\alpha_1 \leq \binom{l}{2} \frac{r-1}{mr-1} \leq \frac{l^2}{2m} = \frac{lr}{2n}$$

where we use the fact that $(a-1)/(b-1) \leq a/b$ for all $a \leq b$ and the relation $mr = nl$. Next, define $\beta_2 \equiv \mathbb{P}[\mathcal{B}_{\boldsymbol{w},2} \text{ is not a tree} \,|\, \mathcal{B}_{\boldsymbol{w},1} \text{ is a tree}]$ so that we have:

$$\mathbb{P}\left(\mathcal{B}_{\boldsymbol{w},2} \text{ is not a tree}\right) \leq \alpha_1 + \beta_2$$

We can similarly bound β_2. For generating $\mathcal{B}_{\boldsymbol{w},2}$ conditioned on $\mathcal{B}_{\boldsymbol{w},1}$ being a tree, there are $l\hat{r}$ half-edges where $\hat{r} = r - 1$. Among the $\binom{l\hat{r}}{2}$ pairs of these half-edges, each pair will be connected to the same worker with probability $\frac{l-1}{l\cdot(n-1)-1}$ and therefore:

$$\begin{aligned}\beta_2 &\leq \frac{l^2\hat{r}^2}{2} \frac{l-1}{l\cdot(n-1)-1} \\ &\leq \frac{l^2\hat{r}^2}{2} \frac{l}{l\cdot(n-1)} = \frac{l^2\hat{r}^2}{2(n-1)}\end{aligned}$$

Combining this with the above, we get that

$$\mathbb{P}[\mathcal{B}_{\boldsymbol{w},2} \text{ is not a tree}] \leq \alpha_1 + \beta_2 \leq \frac{lr}{2n} + \frac{l^2\hat{r}^2}{2(n-1)} \leq \frac{l^2r^2}{n-1}$$

B.2.2 Proof of Lemma 2.6

Consider an honest worker w_i with a given reliability μ_i. Recall that the penalty assigned to w_i is of the form:

$$\mathbf{PEN}_i = \frac{1}{l} \sum_{j \in \mathcal{T}_i} \boldsymbol{S}_{ij},$$

where $\boldsymbol{S}_{ij} = \frac{1}{\boldsymbol{D}_j^+}$ if $\boldsymbol{W}_{ij} = +1$ and $\boldsymbol{S}_{ij} = \frac{1}{\boldsymbol{D}_j^-}$ when $\boldsymbol{W}_{ij} = -1$. For any two tasks $t_j \neq t_{j'} \in \mathcal{T}_i$, we claim that $\boldsymbol{W}_{ij}$ and $\boldsymbol{W}_{ij'}$ are independent, conditioned on the reliability $\boldsymbol{M}_i$. This is

because for $(v_1, v_2) \in \{-1, +1\}$, we can show:

$$\begin{aligned}
&\mathbb{P}[\boldsymbol{W}_{ij} = v_1; \boldsymbol{W}_{ij'} = v_2 \,|\, \boldsymbol{M}_i = \mu_i] \\
&= \sum_{(x_1,x_2)\in\{-1,+1\}} \mathbb{P}[\boldsymbol{W}_{ij} = v_1; \boldsymbol{W}_{ij'} = v_2 \,|\, \boldsymbol{M}_i = \mu_i; \boldsymbol{Y}_j = x_1; \boldsymbol{Y}_{j'} = x_2] \cdot \mathbb{P}[\boldsymbol{Y}_j = x_1; \boldsymbol{Y}_{j'} = x_2] \\
&\text{(since the true task labels are independent of } \boldsymbol{M}_i) \\
&= \Big(\mathbb{P}[\boldsymbol{W}_{ij} = v_1 \,|\, \boldsymbol{M}_i = \mu_i; \boldsymbol{Y}_j = +1] \cdot \mathbb{P}[\boldsymbol{Y}_j = +1] + \mathbb{P}[\boldsymbol{W}_{ij} = v_1 \,|\, \boldsymbol{M}_i = \mu_i; \boldsymbol{Y}_j = -1] \cdot \mathbb{P}[\boldsymbol{Y}_j = -1]\Big) \cdot \\
&\Big(\mathbb{P}[\boldsymbol{W}_{ij'} = v_2 \,|\, \boldsymbol{M}_i = \mu_i; \boldsymbol{Y}_{j'} = +1] \cdot \mathbb{P}[\boldsymbol{Y}_{j'} = +1] + \mathbb{P}[\boldsymbol{W}_{ij'} = v_2 \,|\, \boldsymbol{M}_i = \mu_i; \boldsymbol{Y}_{j'} = -1] \cdot \mathbb{P}[\boldsymbol{Y}_{j'} = -1]\Big) \\
&= \mathbb{P}[\boldsymbol{W}_{ij} = v_1 \,|\, \boldsymbol{M}_i = \mu_i] \cdot \mathbb{P}[\boldsymbol{W}_{ij'} = v_2 \,|\, \boldsymbol{M}_i = \mu_i]
\end{aligned}$$

Note that the third equality makes use of the fact that $\boldsymbol{W}_{ij}$ (resp. $\boldsymbol{W}_{ij'}$) is independent of all other random variables conditioned on the reliability $\boldsymbol{M}_i$ and the true label $\boldsymbol{Y}_j$ (resp. $\boldsymbol{Y}_{j'}$). The argument above can be extended for any subset of random variables $\boldsymbol{W}_{ij_1}, \boldsymbol{W}_{ij_2}, \ldots, \boldsymbol{W}_{ij_i}$. Further, if the worker-task assignment graph is 2-locally tree-like at w_i, there is no other overlap in the set of workers labeling the tasks $t_j, t_{j'}$ apart from w_i. This combined with the above claim statement shows that the random variables $\{\boldsymbol{S}_{ij} : \ j \in \mathcal{T}_i\}$ are *mutually independent* under our generative model. In addition, since $\frac{1}{r} \leq \boldsymbol{S}_{ij} \leq 1$ for any task $t_j \in \mathcal{T}_i$, i.e the random variables $\boldsymbol{S}_{ij}$ are bounded, we can apply Hoeffding's inequality to bound the difference between $\mathbf{PEN}_i$ and $\mathbb{E}[\mathbf{PEN}_i \,|\, \boldsymbol{M}_i = \mu_i]$ for any $\varepsilon > 0$:

$$\mathbb{P}\left(\mathbf{PEN}_i \geq \mathbb{E}[\mathbf{PEN}_i \mid \boldsymbol{M}_i = \mu_i] + \varepsilon \,\middle|\, \boldsymbol{M}_i = \mu_i\right) \leq \exp\left(\frac{-2l\varepsilon^2}{(1-1/r)^2}\right)$$

The result then follows.

B.2.3 Proof of Theorem 2.4

Suppose we draw a random worker $\boldsymbol{w}$ uniformly from the set of workers W. Then, we want to compute the average error probability, which is the probability that we misclassify a randomly chosen worker:

$$\frac{1}{n}\sum_{i=1}^{n} \mathbb{P}\left(\boldsymbol{I}(w_i) \neq \hat{\boldsymbol{I}}_\theta(w_i) \text{ and } \boldsymbol{M}_i > \hat{\mu}(\theta)\right) = \mathbb{P}\left(\boldsymbol{I}(\boldsymbol{w}) \neq \hat{\boldsymbol{I}}_\theta(\boldsymbol{w}) \text{ and } \boldsymbol{M}_{\boldsymbol{w}} > \hat{\mu}(\theta)\right)$$

Let $\mathbf{PEN}_{\boldsymbol{w}}$ denote the penalty received by the worker $\boldsymbol{w}$. Recall the expression for the expected penalty received by an honest worker from the proof of Theorem 2.2 above:

$$\mathbb{E}[\mathbf{PEN}_{\boldsymbol{w}} \,|\, \boldsymbol{M}_{\boldsymbol{w}} = \mu_{\boldsymbol{w}}] = g(\mu) - \mu_{\boldsymbol{w}} \cdot (g(\mu) - g(1-\mu)).$$

Based on the definition of $\hat{\mu}(\theta)$ in the theorem it follows that

$$p_{\hat{\mu}(\theta)} := \mathbb{E}[\mathbf{PEN}_{\boldsymbol{w}} \,|\, \boldsymbol{M}_{\boldsymbol{w}} = \hat{\mu}(\theta)] = \theta - \varepsilon.$$

We upper bound the probability $\mathbb{P}\left(\boldsymbol{I}(\boldsymbol{w}) \neq \hat{\boldsymbol{I}}_\theta(\boldsymbol{w}) \text{ and } \boldsymbol{M}_{\boldsymbol{w}} > \hat{\mu}(\theta)\right)$ in two steps. First, if $\mathcal{B}_{\boldsymbol{w},2}$ is not a tree, we suppose that the classifier always declares w_i as adversarial, thereby making an error. So, supposing $\mathcal{B}_{\boldsymbol{w},2}$ is a tree, the probability that we misclassify $\boldsymbol{w}$ is given by

$$
\begin{aligned}
&\mathbb{P}(\mathbf{PEN}_{\boldsymbol{w}} > \theta \text{ and } \boldsymbol{M}_{\boldsymbol{w}} > \hat{\mu}(\theta)) \\
&= \mathbb{P}(\mathbf{PEN}_{\boldsymbol{w}} > p_{\hat{\mu}(\theta)} + \varepsilon \text{ and } \boldsymbol{M}_{\boldsymbol{w}} > \hat{\mu}(\theta)) \\
&= \int_{\hat{\mu}(\theta)}^{1} \mathbb{P}\left(\mathbf{PEN}_{\boldsymbol{w}} > p_{\hat{\mu}(\theta)} + \varepsilon \,|\, \boldsymbol{M}_{\boldsymbol{w}} = \tilde{\mu}\right) f(\tilde{\mu}) d\tilde{\mu} \\
&\leq \int_{\hat{\mu}(\theta)}^{1} \mathbb{P}\left(\mathbf{PEN}_{\boldsymbol{w}} > \mathbb{E}[\mathbf{PEN}_{\boldsymbol{w}} \mid \boldsymbol{M}_{\boldsymbol{w}} = \tilde{\mu}] + \varepsilon \,|\, \boldsymbol{M}_{\boldsymbol{w}} = \tilde{\mu}\right) f(\tilde{\mu}) d\tilde{\mu} \\
&(\text{since } \tilde{\mu} > \hat{\mu}(\theta) \implies \mathbb{E}[\mathbf{PEN}_{\boldsymbol{w}} \mid \boldsymbol{M}_{\boldsymbol{w}} = \tilde{\mu}] < p_{\hat{\mu}(\theta)}) \\
&\leq \int_{\hat{\mu}(\theta)}^{1} \exp\left(\frac{-2l\varepsilon^2}{(1-1/r)^2}\right) f(\tilde{\mu}) d\tilde{\mu} \\
&(\text{from Lemma 2.6}) \\
&\leq \exp\left(\frac{-2l\varepsilon^2}{(1-1/r)^2}\right)
\end{aligned}
$$

Then, combining the two cases, it follows that

$$
\begin{aligned}
&\frac{1}{n}\sum_{i=1}^{n} \mathbb{P}\left(\boldsymbol{I}(w_i) \neq \hat{\boldsymbol{I}}_\theta(w_i) \text{ and } \boldsymbol{M}_i > \hat{\mu}(\theta)\right) \\
&= \mathbb{P}\left(\boldsymbol{I}(\boldsymbol{w}) \neq \hat{\boldsymbol{I}}_\theta(\boldsymbol{w}) \text{ and } \boldsymbol{M}_{\boldsymbol{w}} > \hat{\mu}(\theta)\right) \\
&\leq \mathbb{P}(\mathcal{B}_{\boldsymbol{w},2} \text{ is not a tree }) \cdot 1 + \mathbb{P}\left(\boldsymbol{I}(\boldsymbol{w}) \neq \hat{\boldsymbol{I}}_\theta(\boldsymbol{w}) \text{ and } \boldsymbol{M}_{\boldsymbol{w}} > \hat{\mu}(\theta) \,|\, \mathcal{B}_{\boldsymbol{w},2} \text{ is a tree }\right) \cdot 1 \\
&\leq \frac{l^2 r^2}{n-1} + \exp\left(\frac{-2l\varepsilon^2}{(1-1/r)^2}\right)
\end{aligned}
$$

and the first part of the theorem follows. For the second part, when $l = \log n$ and r is fixed, note that

$$
\exp\left(\frac{-2l\varepsilon^2}{(1-1/r)^2}\right) = \exp\left(\frac{-2\log n \cdot \varepsilon^2}{(1-1/r)^2}\right) = \exp\left(\log\left(\frac{1}{n}\right)^{\frac{2\varepsilon^2}{(1-1/r)^2}}\right) = O\left(\frac{1}{n^{2\varepsilon^2}}\right)
$$

In addition, because $\varepsilon < \frac{1}{\sqrt{2}} \implies 2\varepsilon^2 < 1$, it follows that $\frac{\log^2 n \cdot r^2}{n-1} = O\left(\frac{1}{n^{2\varepsilon^2}}\right)$ and the claim follows.

B.2.4 Proof of Theorem 2.7

We follow a similar line of reasoning to the proof of Theorem 2.4. As before, if $\mathcal{B}_{\boldsymbol{w},2}$ is not a tree, we suppose that the worker is always misclassified.

First, we focus on **honest** workers. Given a threshold θ, choose the reliability threshold $\hat{\mu}(q,\theta)$ such that

$$\mathbb{E}[\mathbf{PEN}_i \mid \boldsymbol{M}_i = \hat{\mu}(q,\theta)] = \theta - \varepsilon,$$

where the expected penalty for an honest worker with a fixed reliability was computed in Lemma B.2. Specifically, we have the following expression for $\hat{\mu}(q,\theta)$:

$$\begin{aligned}
&\gamma \cdot \Big(\hat{\mu}(q,\theta)\cdot g(1-P) + (1-\hat{\mu}(q,\theta))\cdot g(P)\Big) + (1-\gamma)\cdot\Big(\hat{\mu}(q,\theta)\cdot g(Q) + (1-\hat{\mu}(q,\theta))\cdot g(1-Q)\Big)\\
&= \theta - \varepsilon\\
&\implies\\
&\Big(\gamma\cdot g(1-P) - \gamma\cdot g(P) + (1-\gamma)\cdot g(Q) - (1-\gamma)\cdot g(1-Q)\Big)\cdot\hat{\mu}(q,\theta)\\
&= \theta - \varepsilon - \gamma\cdot g(P) - (1-\gamma)\cdot g(1-Q)\\
&\implies\\
&\hat{\mu}(q,\theta) = \frac{\gamma\cdot g(P) + (1-\gamma)\cdot g(1-Q) + \varepsilon - \theta}{\gamma\cdot(g(P)-g(1-P)) + (1-\gamma)\cdot(g(1-Q)-g(Q))}
\end{aligned}$$

Note that such a threshold always exists since (1) $\theta - \varepsilon > p_h = \mathbb{E}[\mathbf{PEN}_i \mid \boldsymbol{M}_i = \mu]$, (2) $\mathbb{E}[\mathbf{PEN}_i \mid \boldsymbol{M}_i = \mu_i]$ (strictly) increases as μ_i decreases, and (3) $\mathbb{E}[\mathbf{PEN}_i \mid \boldsymbol{M}_i = 0] \geq p_a > \theta > \theta - \varepsilon$. In particular, this means that $\hat{\mu}(q,\theta) < \mu$. Further, observe that if $q = 1$, then we have $\hat{\mu}(1,\theta) = \hat{\mu}(\theta)$.

Then, the probability that we misclassify a randomly chosen honest worker is given by

$$\mathbb{P}(\mathbf{PEN}_{\boldsymbol{w}} > \theta \,|\, \boldsymbol{w} \text{ is honest }) \leq \mathbb{P}\Big(\mathbf{PEN}_{\boldsymbol{w}} \geq \mathbb{E}[\mathbf{PEN}_{\boldsymbol{w}} \mid \boldsymbol{M}_{\boldsymbol{w}} = \hat{\mu}(q,\theta)] + \varepsilon \,|\, \boldsymbol{w} \text{ is honest }\Big)$$

The claim then follows from the result of Lemma 2.8 below.

When $\boldsymbol{w}$ is **adversarial**, the probability that we misclassify $\boldsymbol{w}$ is given by

$$\mathbb{P}(\mathbf{PEN}_{\boldsymbol{w}} \leq \theta \mid \boldsymbol{w} \text{ is adversarial}) \leq \mathbb{P}(\mathbf{PEN}_{\boldsymbol{w}} \leq p_a - \varepsilon \mid \boldsymbol{w} \text{ is adversarial}) \leq \exp\left(\frac{-2l\varepsilon^2}{(1-1/r)^2}\right)$$

where the first inequality follows since $\theta < p_a - \varepsilon$ and the second follows from the result of Lemma 2.8 below.

Coming to the second part of the theorem, first observe that the expected penalties p_h and p_a lie between 0 and 1. As a result, we have that $\varepsilon < (p_a - p_h)/2 \leq \frac{1}{2} \implies 2\varepsilon^2 < 1$ and the claim follows from the sequence of arguments in the proof of Theorem 2.4 above.

B.2.5 Proof of Lemma 2.8

Suppose that worker w_i is **honest** and let $p_{\hat{\mu}} = \mathbb{E}[\mathbf{PEN}_i \,|\, \boldsymbol{M}_i = \hat{\mu}]$. We have the following (conditioned on $\mathcal{B}_{w_i,2}$ being a tree):

$$
\begin{aligned}
&\mathbb{P}(\mathbf{PEN}_i \geq p_{\hat{\mu}} + \epsilon \,|\, w_i \text{ is honest }) \\
&= \int_0^1 \mathbb{P}(\mathbf{PEN}_i \geq p_{\hat{\mu}} + \epsilon \,|\, w_i \text{ is honest and } \boldsymbol{M}_i = \mu_i) f(\mu_i) d\mu_i \\
&= \int_0^{\hat{\mu}} \mathbb{P}(\mathbf{PEN}_i \geq p_{\hat{\mu}} + \epsilon \,|\, w_i \text{ is honest and } \boldsymbol{M}_i = \mu_i) f(\mu_i) d\mu_i \\
&+ \int_{\hat{\mu}}^1 \mathbb{P}(\mathbf{PEN}_i \geq p_{\hat{\mu}} + \epsilon \,|\, w_i \text{ is honest and } \boldsymbol{M}_i = \mu_i) f(\mu_i) d\mu_i \\
&\leq \int_0^{\hat{\mu}} f(\mu_i) d\mu_i \\
&+ \int_{\hat{\mu}}^1 \mathbb{P}(\mathbf{PEN}_i \geq \mathbb{E}[\mathbf{PEN}_i \mid \boldsymbol{M}_i = \mu_i] + \epsilon \,|\, w_i \text{ is honest and } \boldsymbol{M}_i = \mu_i) f(\mu_i) d\mu_i \\
&(\text{since } \mu_i \geq \hat{\mu} \implies \mathbb{E}[\mathbf{PEN}_i \mid \boldsymbol{M}_i = \mu_i] \leq p_{\hat{\mu}}) \\
&\leq F(\hat{\mu}) + \exp\left(\frac{-2l\varepsilon^2}{(1-1/r)^2}\right) \\
&(\text{from Lemma 2.6})
\end{aligned}
$$

When w_i is an **adversary**, note that its responses $\{\boldsymbol{W}_{ij} : j \in \mathcal{T}_i\}$ are trivially independent since adversarial workers always respond with $+1$ on all assigned tasks. Further, since $\mathcal{B}_{w_i,2}$ is locally tree-like, there is no other overlap in the set of workers labeling any two tasks $t_j, t_{j'} \in \mathcal{T}_i$ apart from w_i. As a result, the assigned penalties $\{\boldsymbol{S}_{ij} : j \in \mathcal{T}_i\}$ are *mutually independent* and using the Hoeffding's inequality, we can establish the concentration bound for adversarial workers:

$$
\mathbb{P}[\mathbf{PEN}_i \leq p_a - \varepsilon \,|\, w_i \text{ is adversarial}] \leq \exp\left(\frac{-2l^2\varepsilon^2}{\sum_{j=1}^{l}(1-1/r)^2}\right) = \exp\left(\frac{-2l\varepsilon^2}{(1-1/r)^2}\right)
$$

B.2.6 Extending results to random, normalized variant of hard-penalty algorithm

Here, we show that the theoretical results proved above for the soft-penalty algorithm extend to the random, normalized variant of the hard-penalty algorithm mentioned in Section 2.3.3. First, we focus on the expected penalties. Since the penalty algorithm is randomized, the expectation also takes into account the randomness in the algorithm. As above, if $\boldsymbol{S}_{ij}$ denotes the penalty received by worker w_i from task t_j, then we have that $\boldsymbol{S}_{ij} \in \{0, 1\}$. Further, conditioned on the fact that $\boldsymbol{W}_{ij} = +1$, we have that $\mathbb{E}[\boldsymbol{S}_{ij} \mid \boldsymbol{W}_{ij} =$

$+1] = \mathbb{E}[\frac{1}{\boldsymbol{D}_j^+} \mid \boldsymbol{W}_{ij} = +1]$ using the law of iterated expectations. Similarly, for the case when $\boldsymbol{W}_{ij} = -1$. Then, using the arguments above it is easy to see that the expressions for the expected penalties are the same.

Moving on to the concentration results, observe that $\boldsymbol{S}_{ij}$ depends *only* on $\boldsymbol{W}_{ij}$ and $\boldsymbol{D}_j^+$ (and consequently, $\boldsymbol{D}_j^-$). This is because when $\boldsymbol{W}_{ij} = +1$, we have

$$(\boldsymbol{S}_{ij} \mid \boldsymbol{W}_{ij} = +1) = \begin{cases} 1 & \text{w.p. } 1/\boldsymbol{D}_j^+ \\ 0 & \text{w.p. } 1 - 1/\boldsymbol{D}_j^+ \end{cases}$$

Similarly, for the case when $\boldsymbol{W}_{ij} = -1$. Now, when $\mathcal{B}$ is locally tree-like at worker node w_i, the arguments in the proofs of Lemma 2.6 and 2.8 establish that the random variables $\{\boldsymbol{S}_{ij} \ : \ t_j \in \mathcal{T}_i\}$ are still mutually independent conditioned on the identity of worker w_i (this also relies on the fact that each task is treated independently when computing the random semi-matching). Therefore, we can apply the Hoeffding's bound to establish the concentration of the penalties under the random, normalized variant of the hard-penalty algorithm around the expected values.

B.3 Proof of Lemma 2.9

For the simple majority algorithm, it is easy to see that each task has k incorrect responses and at most $r < k$ correct responses and consequently, it outputs the incorrect label for all tasks.

For the expectation-maximization (EM) algorithm, recall that the generative model for worker ratings is as follows: the true task labels are sampled from a population with $\gamma \in [0, 1]$ fraction of tasks having $+1$ true label. Each worker w_i has reliability parameter $\mu_i \in [0, 1]$, which specifies the probability that w_i provides the correct label on any task. Then, given the response matrix $\mathcal{L}$, the log-likelihood of the parameters, $\boldsymbol{\Theta} = (\mu_1, \mu_2, \cdots, \mu_n, \gamma)$ under this generative model can be written as:

$$\log \mathbb{P}[\mathcal{L} \mid \boldsymbol{\Theta}] = \sum_{j=1}^{m} \log \left(a_j(\boldsymbol{\mu}) \cdot \gamma + b_j(\boldsymbol{\mu}) \cdot (1 - \gamma) \right)$$

where $a_j(\boldsymbol{\mu}), b_j(\boldsymbol{\mu})$ are defined as (recall that W_j is the set of workers who label task t_j)

$$a_j(\boldsymbol{\mu}) = \prod_{i \in W_j} \mu_i^{\mathbf{1}[\mathcal{L}_{ij} = +1]} \cdot (1 - \mu_i)^{\mathbf{1}[\mathcal{L}_{ij} = -1]}$$

$$b_j(\boldsymbol{\mu}) = \prod_{i \in W_j} (1 - \mu_i)^{\mathbf{1}[\mathcal{L}_{ij} = +1]} \cdot \mu_i^{\mathbf{1}[\mathcal{L}_{ij} = -1]}$$

The objective is to find the model parameters $\boldsymbol{\Theta}^*$ that maximize the log-likelihood, i.e. $\boldsymbol{\Theta}^* = \arg\max_{\boldsymbol{\Theta}} \log \mathbb{P}[\mathcal{L} \mid \boldsymbol{\Theta}]$. The EM algorithm computes the maximum likelihood estimates

(MLE) of the model parameters by introducing the latent true label of each task, denoted by the vector $\boldsymbol{y} = [y_1, y_2, \ldots, y_m]$. The complete data log-likelihood can then be written as:

$$\log \mathbb{P}[\mathcal{L}, \boldsymbol{y} \mid \boldsymbol{\Theta}] = \sum_{j=1}^{m} \Big(y_j \log(a_j(\boldsymbol{\mu})\gamma) + (1 - y_j) \log\left(b_j(\boldsymbol{\mu}) \cdot (1-\gamma)\right) \Big) \tag{B.8}$$

Starting from an initial estimate $\boldsymbol{\Theta}^{(0)}$, each iteration $x \geq 1$ of the EM algorithm consists of two steps:

- **E-step:** Given the response matrix $\mathcal{L}$ and the current estimate of the model parameters $\boldsymbol{\Theta}^{(x-1)}$, the expectation of the complete data log-likelihood (w.r.t to the conditional distribution $\boldsymbol{y} \mid \mathcal{L}; \boldsymbol{\Theta}^{(x-1)}$) is computed as

$$\mathbb{E}_{\boldsymbol{y} \mid \mathcal{L}; \boldsymbol{\Theta}^{(x-1)}} \{\log \mathbb{P}[\mathcal{L}, \boldsymbol{y} \mid \boldsymbol{\Theta}]\} = \sum_{j=1}^{m} \Big(\delta_j^{(x-1)} \log(a_j(\boldsymbol{\mu}) \cdot \gamma) + (1 - \delta_j^{(x-1)}) \log\left(b_j(\boldsymbol{\mu}) \cdot (1-\gamma)\right) \Big)$$

  where $\delta_j^{(x-1)} = \mathbb{P}[y_j = +1 \mid \mathcal{L}; \boldsymbol{\Theta}^{(x-1)}]$ is the current posterior estimate of the true label of task t_j being $+1$. Using Bayes theorem, we can compute

$$\begin{aligned} \delta_j^{(x-1)} &= \frac{\mathbb{P}[\{\mathcal{L}_{ij} : \ i \in W_j\} \mid y_j = +1; \boldsymbol{\Theta}^{(x-1)}] \cdot \mathbb{P}[y_j = +1 \mid \boldsymbol{\Theta}^{(x-1)}]}{\mathbb{P}[\{\mathcal{L}_{ij} : \ i \in W_j\} \mid \boldsymbol{\Theta}^{(x-1)}]} \\ &= \frac{a_j(\boldsymbol{\mu}^{(x-1)}) \cdot \gamma^{(x-1)}}{a_j(\boldsymbol{\mu}^{(x-1)}) \cdot \gamma^{(x-1)} + b_j(\boldsymbol{\mu}^{(x-1)}) \cdot (1 - \gamma^{(x-1)})} \end{aligned} \tag{B.9}$$

  where the first equality follows from the fact that labels for different tasks t_j are independent of each other and the second equality follows from the law of total probability.

- **M-step:** Based on the current posterior estimates of the true labels $\delta_j^{(x-1)}$, the model parameters are updated by maximizing the expected complete data log-likelihood $\mathbb{E}_{\boldsymbol{y} \mid \mathcal{L}; \boldsymbol{\Theta}^{(x-1)}} \{\log \mathbb{P}[\mathcal{L}, \boldsymbol{y} \mid \boldsymbol{\Theta}]\}$, which can be shown to be a lower bound on the (incomplete) data log-likelihood $\log \mathbb{P}[\mathcal{L} \mid \boldsymbol{\Theta}]$. The prevalence of positive tasks γ is updated as:

$$\gamma^{(x)} = \frac{\sum_{j=1}^{m} \delta_j^{(x-1)}}{m} \tag{B.10}$$

  Similarly, the reliability of worker w_i is updated as:

$$\mu_i^{(x)} = \frac{\sum_{j \in \mathcal{T}_i} \Big(\mathbf{1}[\mathcal{L}_{ij} = +1] \cdot \delta_j^{(x-1)} + \mathbf{1}[\mathcal{L}_{ij} = -1] \cdot (1 - \delta_j^{(x-1)}) \Big)}{|\mathcal{T}_i|} \tag{B.11}$$

  where recall that $\mathcal{T}_i$ is the set of tasks assigned to worker w_i.

These two steps are iterated until convergence of the data log-likelihood $\log \mathbb{P}[\mathcal{L} \mid \boldsymbol{\Theta}]$. After convergence, the true labels of the tasks are estimated as:

$$\hat{y}_j = 2 \cdot \mathbf{1}[\delta_j^{(K)} \geq 0.5] - 1 \tag{B.12}$$

where K is the number of iterations to convergence.

Since the original problem is non-convex, the EM algorithm converges to a local optimum in general. Its performance critically depends on the initialization of the posterior estimates $\delta_j^{(0)}$ for each task t_j. A popular approach is to use the majority estimate (see [RY12]), given by $\delta_j^{(0)} := \frac{\sum_{i \in W_j} \mathbf{1}[\mathcal{L}_{ij}=+1]}{|W_j|}$. We show that this initialization results in incorrect labels for all the tasks.

Suppose that the true labels for all tasks are $+1$, i.e $y_j = +1$ for all $1 \leq j \leq m$. Then all honest worker labels are $+1$ and adversary labels are -1. Further, since each task has more adversary than honest labels, we have that $\delta_j^{(0)} < 0.5$ for all $1 \leq j \leq m$. Given this, it easy to see from equation (B.11) that $\mu_i^{(1)} < 0.5$ whenever w_i is honest and $\mu_i^{(1)} > 0.5$ for all adversarial workers w_i. In addition, we have $\gamma^{(1)} < 0.5$ which follows from equation (B.10). With these estimates, we update the posterior probabilities δ_j for each task t_j. Again, it follows that $\delta_j^{(1)} < 0.5$ for all $1 \leq j \leq m$, using the update rule in equation (B.9). The above sequence of claims shows that $\delta_j^{(x-1)} < 0.5$ for all iterations $x \geq 1$ and consequently, $\hat{y}_j = -1$ for all tasks t_j (using equation (B.12)). Therefore, the EM algorithm outputs incorrect labels for all tasks.

Now, let us consider the general case. If we initialize using the majority estimate, we obtain that $\delta_j^{(0)} > 0.5$ for all tasks with true label $y_j = -1$ and $\delta_j^{(0)} < 0.5$ for all tasks with $y_j = +1$. Then, it follows from equation (B.11) that $\mu_i^{(1)} < 0.5$ for all honest workers w_i and $\mu_i^{(1)} > 0.5$ for adversarial workers w_i. In addition, if we look at the update equations (B.10) and (B.11) together, we can also show that $\mu_i^{(1)} \geq \max(1-\gamma^{(1)}, \gamma^{(1)})$ for all adversaries w_i. To see this, first note that adversaries label on all the tasks, so that $|\mathcal{T}_i| = m$ for an adversarial worker w_i. Next, let $\mathcal{T}_i^+$ and $\mathcal{T}_i^-$ denote the set of tasks for which adversary w_i labels $+1$ and -1 respectively. We need to consider different cases based on whether $\mathcal{T}_i^+$ or $\mathcal{T}_i^-$ are empty:

If $\mathcal{T}_i^+$ is empty, then it follows that $\mu_i^{(1)} = 1 - \gamma^{(1)}$. Also, since all adversary labels are -1, we have $\delta_j^{(0)} < 0.5$ for all $1 \leq j \leq m$ and therefore $\gamma^{(1)} < 1 - \gamma^{(1)}$. Therefore, in this case we have $\mu_i^{(1)} \geq \max(1 - \gamma^{(1)}, \gamma^{(1)})$. A symmetric argument can be applied when $\mathcal{T}_i^-$ is empty.

So suppose that both $\mathcal{T}_i^+$ and $\mathcal{T}_i^-$ are non-empty, then we have from equation (B.11):

$$\begin{aligned}
\mu_i^{(1)} &= \frac{\sum_{j \in \mathcal{T}_i} \left(\mathbf{1}[\mathcal{L}_{ij} = +1] \cdot \delta_j^{(0)} + \mathbf{1}[\mathcal{L}_{ij} = -1] \cdot (1 - \delta_j^{(0)}) \right)}{|\mathcal{T}_i|} \\
&= \frac{\sum_{j \in \mathcal{T}_i^+} \delta_j^{(0)} + \sum_{j \in \mathcal{T}_i^-} (1 - \delta_j^{(0)})}{m} \\
&> \frac{\sum_{j \in \mathcal{T}_i^+} \delta_j^{(0)} + \sum_{j \in \mathcal{T}_i^-} \delta_j^{(0)}}{m} \\
&\text{(since } \delta_j^{(0)} < 0.5 \text{ if adversaries label } t_j \text{ as } -1) \\
&= \frac{\sum_{j \in \mathcal{T}_i} \delta_j^{(0)}}{m} \\
&= \gamma^{(1)}
\end{aligned}$$

where the last equality follows from equation (B.10). Similarly, we can show that $\mu_i^{(1)} > 1 - \gamma^{(1)}$ and the claim follows.

Now, let $W_j^{(h)}$ and $W_j^{(a)}$ denote the set of honest and adversarial workers labeling task t_j. Consider a task t_j for which the true label $y_j = -1$. Now, we have two cases:

Case 1: $\gamma^{(1)} \geq 0.5$. First observe that since $\mu_i^{(1)} < 0.5$ for all honest workers w_i, we have $\mu_i^{(1)} < 1 - \mu_i^{(1)}$ for all honest workers w_i. Similarly, we have $1 - \mu_i^{(1)} < \mu_i^{(1)}$ for all adversarial workers w_i. Then, we have

$$\begin{aligned}
b_j(\boldsymbol{\mu}^{(1)}) \cdot (1 - \gamma^{(1)}) &= \left(\prod_{i \in W_j} (1 - \mu_i^{(1)})^{\mathbf{1}[\mathcal{L}_{ij} = +1]} \cdot (\mu_i^{(1)})^{\mathbf{1}[\mathcal{L}_{ij} = -1]} \right) \cdot (1 - \gamma^{(1)}) \\
&= \prod_{i \in W_j^{(h)}} \mu_i^{(1)} \cdot \prod_{i \in W_j^{(a)}} (1 - \mu_i^{(1)}) \cdot (1 - \gamma^{(1)}) \\
&\text{(since honest worker response is } -1 \text{ and adversary response is } +1) \\
&< \prod_{i \in W_j^{(h)}} (1 - \mu_i^{(1)}) \cdot \prod_{i \in W_j^{(a)}} \mu_i^{(1)} \cdot \gamma^{(1)} \\
&= a_j(\boldsymbol{\mu}^{(1)}) \cdot \gamma^{(1)}
\end{aligned}$$

where the inequality follows from the observation above and the fact that $\gamma^{(1)} \geq 0.5$. Using equation (B.9), it follows that $\delta_j^{(1)} > 0.5$.

Case 2: $\gamma^{(1)} < 0.5$. In this scenario, we have:

$$
\begin{aligned}
& b_j(\boldsymbol{\mu}^{(1)}) \cdot (1 - \gamma^{(1)}) \\
&= \left(\prod_{i \in W_j} (1 - \mu_i^{(1)})^{\mathbf{1}[\mathcal{L}_{ij} = +1]} \cdot (\mu_i^{(1)})^{\mathbf{1}[\mathcal{L}_{ij} = -1]} \right) \cdot (1 - \gamma^{(1)}) \\
&= \prod_{i \in W_j^{(h)}} \mu_i^{(1)} \cdot \prod_{i \in W_j^{(a)}} (1 - \mu_i^{(1)}) \cdot (1 - \gamma^{(1)}) \\
&= \prod_{i \in W_j^{(h)}} \mu_i^{(1)} \cdot (1 - \mu_{w_{i_1}}^{(1)}) \cdot (1 - \gamma^{(1)}) \cdot \prod_{i \in W_j^{(a)} \setminus \{i_1\}} (1 - \mu_i^{(1)}) \\
& \text{(where } w_{i_1} \text{ is some adversarial worker)} \\
&\leq \prod_{i \in W_j^{(h)}} \mu_i^{(1)} \cdot \mu_{w_{i_1}}^{(1)} \cdot \gamma^{(1)} \cdot \prod_{i \in W_j^{(a)} \setminus \{i_1\}} (1 - \mu_i^{(1)}) \\
& \Big(\text{since } 1 - \gamma^{(1)} \leq \mu_{w_{i_1}}^{(1)} \text{ because } w_{i_1} \text{ is adversarial; see discussion before Case 1 above} \Big) \\
&< \prod_{i \in W_j^{(h)}} (1 - \mu_i^{(1)}) \cdot \mu_{w_{i_1}}^{(1)} \cdot \gamma^{(1)} \cdot \prod_{i \in W_j^{(a)} \setminus \{i_1\}} \mu_i^{(1)} \\
& \text{(from Case 1 above)} \\
&= a_j(\boldsymbol{\mu}^{(1)}) \cdot \gamma^{(1)}
\end{aligned}
$$

Consequently, in both cases, the posterior probability $\delta_j^{(1)} > 0.5$ given that the true label was $y_j = -1$. A symmetric argument shows that $\delta_j^{(1)} < 0.5$ if $y_j = +1$. Then, the above sequence of claims establishes that this remains true for all iterations in the future. Finally, step (B.12) implies that the output label is incorrect for all tasks.

B.4 Proof of Theorem 2.10

We prove the result for the case when there exists at least one subset $\mathcal{T}' \subseteq \mathcal{T}$ such that $\text{PreIm}(\mathcal{T}') \leq k$. Otherwise, the lower bound $L = 0$ by definition and the result of the theorem is trivially true.

Let H^* denote the set $\text{PreIm}(\mathcal{T}^*)$ where

$$
\mathcal{T}^* \stackrel{\text{def}}{=} \operatorname*{arg\,max}_{\mathcal{T}' \subseteq \mathcal{T} \colon |\text{PreIm}(\mathcal{T}')| \leq k} |\mathcal{T}'| \,.
$$

We construct an adversary strategy σ^* under which at least L tasks are affected for some true labeling of the tasks, for any decision rule $\boldsymbol{R} \in \mathcal{C}$. Specifically for a fixed honest worker-task assignment graph $\mathcal{B}_H$ and ground-truth labeling $\boldsymbol{y}$ of the tasks, consider the following adversary strategy (that depends on the obtained honest worker responses): letting

$H^* = \{h_1, h_2, \ldots, h_{|H^*|}\}$ and the set of adversaries $A = \{a_1, a_2, \ldots, a_k\}$, we have (recall the notation in Section 2.2)

$$a_i(t) = \begin{cases} -h_i(t) & \text{if } t \in \mathcal{T}^* \\ h_i(t) & \text{otherwise} \end{cases} \quad \forall i = 1, 2, \ldots, |H^*| \tag{B.13}$$

In other words, the adversaries *flip* the labels of the honest workers H^* for tasks in $\mathcal{T}^*$ and *copy* their responses for all other tasks. Note that since $|H^*| \leq k$ by construction, the above strategy is feasible. In addition, if $|H^*| < k$, then we only use $|H^*|$ of the k adversary identities. Let $\mathcal{L}(\boldsymbol{y})$ denote the $n \times m$ response matrix obtained for this adversary strategy, where we explicitly denote the dependence on the true label vector $\boldsymbol{y}$. Here, recall that, n denotes the total number of workers.

Now consider the scenario in which the true labels of all tasks in $\mathcal{T}^*$ were reversed, let this ground-truth be denoted as $\tilde{\boldsymbol{y}}$. Let $\tilde{h}(t)$ denote the response of honest worker h for task t in the new scenario. Since, honest workers always respond correctly, we have that:

$$\tilde{h}(t) = \begin{cases} -h(t) & \text{if } t \in \mathcal{T}^* \\ h(t) & \text{otherwise} \end{cases} \quad \forall\, h \in H \tag{B.14}$$

Correspondingly, according to the adversary labeling strategy σ^* described above, the adversary responses would also change. In particular, using $\tilde{a}_i(t)$ to denote the adversary response in the new scenario, we have

$$\tilde{a}_i(t) = \begin{cases} -\tilde{h}_i(t) & \text{if } t \in \mathcal{T}^* \\ \tilde{h}_i(t) & \text{otherwise} \end{cases} \quad \forall i = 1, 2, \ldots, |H^*| \tag{B.15}$$

Finally, let $\mathcal{L}(\tilde{\boldsymbol{y}})$ denote the response matrix in this new scenario.

We now argue that $\mathcal{L}(\tilde{\boldsymbol{y}}) = \mathcal{P} \cdot \mathcal{L}(\boldsymbol{y})$ for some $n \times n$ permutation matrix $\mathcal{P}$. In order to see this, for any worker w (honest or adversary), let $r(w)$ and $\tilde{r}(w)$ respectively denote the row vectors in matrices $\mathcal{L}(\boldsymbol{y})$ and $\mathcal{L}(\tilde{\boldsymbol{y}})$. We show that $\mathcal{L}(\tilde{\boldsymbol{y}})$ can be obtained from $\mathcal{L}(\boldsymbol{y})$ through a permutation of the rows.

First, for any honest worker $h \notin H^*$, we must have by definition of $\text{PreIm}(\cdot)$ that $h(t) = 0$ for any $t \in \mathcal{T}^*$. Thus, it follows from equation (B.14) that $\tilde{h}(t) = -h(t) = 0 = h(t)$ for any $t \in \mathcal{T}^*$. Furthermore, $\tilde{h}(t) = h(t)$ for any $t \notin \mathcal{T}^*$ by (B.14). Therefore, we have that $\tilde{r}(h) = r(h)$ for any $h \notin H^*$. Next, consider an honest worker $h_i \in H^*$ for some i. It can be argued that $\tilde{r}(h_i) = r(a_i)$. To see this, for any task $t \notin \mathcal{T}^*$, we have by equation (B.14) that $\tilde{h}_i(t) = h_i(t) = a_i(t)$, where the second equality follows from equation (B.13). Similarly, for any $t \in \mathcal{T}^*$, we have $\tilde{h}_i(t) = -h_i(t) = a_i(t)$ again using equation (B.13). Thus, we have shown that the rows $\tilde{r}(h_i) = r(a_i)$ for any $1 \leq i \leq |H^*|$. A symmetric argument shows that $\tilde{r}(a_i) = r(h_i)$ for all $1 \leq i \leq |H^*|$. Consequently, $\mathcal{L}(\tilde{\boldsymbol{y}})$ is obtained from $\mathcal{L}(\boldsymbol{y})$ by swapping rows corresponding to h_i with a_i for all $i = 1, 2, \ldots, |H^*|$.

Next, it follows from the fact that the decision rule $\boldsymbol{R} \in \mathcal{C}$, that $\boldsymbol{R}(\mathcal{L}(\tilde{\boldsymbol{y}})) = \boldsymbol{R}(\mathcal{L}(\boldsymbol{y}))$. Thus, the labels output by $\boldsymbol{R}$ for all tasks in $\mathcal{T}^*$ is the same under both scenarios. As

a result, it follows that $\text{Aff}(\boldsymbol{R}, \sigma^*, \boldsymbol{y}) + \text{Aff}(\boldsymbol{R}, \sigma^*, \tilde{\boldsymbol{y}}) = |\mathcal{T}^*| = 2 * L$ and therefore, either $\text{Aff}(\boldsymbol{R}, \sigma^*, \boldsymbol{y}) \geq L$ or $\text{Aff}(\boldsymbol{R}, \sigma^*, \tilde{\boldsymbol{y}}) \geq L$. In other words, there exists a ground-truth task labeling for which the number of affected tasks is at least L, and since we take a maximum over all possible ground-truth labelings, the result of the theorem follows. Further, any decision rule that breaks ties (i.e equal number of $+1$ and -1 responses) randomly, such as the simple majority rule, will achieve the lower bound L.

Worker misclassification rates. Since we allow the sophisticated adversaries to adopt arbitrary strategies, we need to assume some characteristic property that helps to identify honest workers. In the theorem we assumed that all honest workers are perfectly reliable, so that they agree with each other on their labels for all tasks. Any algorithm that is trying to separate honest workers from the sophisticated adversaries will output two groups of workers, say $W^{(1)}, W^{(2)}$ such that $W^{(1)} \cup W^{(2)} = H \cup A$ and $W^{(1)} \cap W^{(2)} = \phi$. Further, let us suppose that the algorithm has knowledge of k—the number of sophisticated adversaries, so that the output satisfies $\left|W^{(1)}\right| = n - k$ and $\left|W^{(2)}\right| = k$ where $|H \cup A| = n$. We call an output $(W^{(1)}, W^{(2)})$ *valid* if all workers in $W^{(1)}$ agree with each other on their labels for all tasks. Note that the "true" output $W^{(1)} = H$ and $W^{(2)} = A$ is valid. We argue that for the above adversary strategy, the output $W^{(1)} = (H \setminus H^*) \cup A$ and $W^{(2)} = H^*$ is also valid, where H^* is as defined in the proof above.

To see this, first note that we can identify the set $\mathcal{T}^*$ of tasks for which there are conflicts, i.e. both $+1$ and -1 labels, and use it to partition the set of workers labeling these tasks into two groups where workers in each group agree on their label for all tasks. Now, one of these groups comprises adversarial workers A and the other honest workers, corresponding to the set H^*. The remaining workers correspond to the set $H \setminus H^*$. Now, if we consider the output $W^{(1)} = (H \setminus H^*) \cup A$, any two workers in $W^{(1)}$ indeed agree with each other on their labels for all tasks. This is because, by the definition of PreIm, workers in $H \setminus H^*$ *only* label tasks in the set $\mathcal{T} \setminus \mathcal{T}^*$. Further, since adversaries agree with honest workers H^* on these tasks (refer to the strategy above) and honest workers are perfectly reliable, this means that adversaries A and honest workers $H \setminus H^*$ also agree with each other on their labels for all tasks in $\mathcal{T} \setminus \mathcal{T}^*$. Consequently, we have that the output $((H \setminus H^*) \cup A, H^*)$ is also *valid*. Further, this is the only other valid output, since honest workers H^* and adversaries A do *not* agree with each other on their labels for tasks in $\mathcal{T}^*$ and therefore, cannot be placed together in the same group.

Finally, since there is no other information about the worker identities, any algorithm cannot distinguish between the two valid outputs described above, so that a random guess will misclassify $2k$ workers with probability $(1/2)$.

B.4.1 Proof of Corollary 2.10.1

We first prove that $2L \geq \lfloor \frac{k}{r} \rfloor$. Consider $\mathcal{T}' \subseteq \mathcal{T}$ such that $|\mathcal{T}'| = \lfloor \frac{k}{r} \rfloor$, note that we can always choose such a $\mathcal{T}'$ since $k < |H| \leq r \cdot |\mathcal{T}| \implies k/r < |\mathcal{T}|$. Since $\mathcal{B}_H$ is r-right regular, the pre-image of $\mathcal{T}'$ in $\mathcal{B}_H$ satisfies $|\text{PreIm}(\mathcal{T}')| \leq r \cdot |\mathcal{T}'| = r \cdot \lfloor \frac{k}{r} \rfloor \leq k$. In other words, any subset of tasks of size $\lfloor \frac{k}{r} \rfloor$ has a pre-image of size at most k. By the definition of L, we have

that $2L \geq \lfloor \frac{k}{r} \rfloor$.

For the upper-bound, consider any $\mathcal{T}' \subset \mathcal{T}$ such that $|\mathcal{T}'| = e$ where $e := \lceil \frac{k}{\alpha} \rceil + 1$. Since $e < \gamma \cdot |\mathcal{T}|$, by the expander property we have that

$$|\text{PreIm}(\mathcal{T}')| \geq \alpha |\mathcal{T}'| = \alpha \cdot e \geq \alpha \cdot (\frac{k}{\alpha} + 1) > k.$$

This means that any subset $\mathcal{T}'$ of size *at least* e has a pre-image of size strictly greater than k, since the size of the pre-image can only increase with the addition of more tasks. Therefore, this implies that $2L \leq \lceil \frac{k}{\alpha} \rceil$.

For the second part of the corollary, refer to Theorem 4.4 in Chapter 4 of [Vad12].

B.5 Proof of Theorem 2.11

Before we can prove the theorem, we need the following definitions and lemmas.

Definition B.1. A bipartite graph $G = (V_1, V_2, E)$ is termed **degenerate** if the following condition is satisfied:

$$|V_1| > |V_2|.$$

Definition B.2. A bipartite graph $G = (V_1, V_2, E)$ is termed **growth** if the following condition is satisfied:

$$\forall \ V \subseteq V_1, \ \ |V| \leq |\text{Img}(V),|$$

where $\text{Img}(V) = \{v_2 \in V_2 \mid \exists v \in V \ \text{ s.t. } (\text{v}, \text{v}_2) \in \text{E}\}$, i.e. the set of neighboring nodes of V.

Lemma B.3. *Any bipartite graph can be decomposed into degenerate and growth sub-graph, where there are cross-edges only between the left nodes of the growth component and the right nodes of the degenerate component.*

Proof. Let $G = (V_1, V_2, E)$ be a given bipartite graph. Define V^* to be the largest subset of V_1 such that $|V^*| > |\text{Img}_G(V^*)|$ where Img_G denotes the image in the graph G. If no such V^* exists then the graph is already growth and we are done. If $V^* = V_1$ then the graph is degenerate and again we are done. Else, we claim that the sub-graph J of G restricted to $V_1 \setminus V^*$ on the left and $V_2 \setminus \text{Img}_G(V^*)$ on the right is growth. Suppose not, then there exists a subset V' of nodes on the left such that $|V'| > |\text{Img}_J(V')|$ where $\text{Img}_J(V') \subseteq V_2 \setminus \text{Img}_G(V^*)$ denotes the image of V' in the sub-graph J. But then, we can add V' to V^* to get a larger degenerate sub-graph in G, contradicting our choice of V^*. Also, note that the only cross-edges are between $V_1 \setminus V^*$ and $\text{Img}_G(V^*)$. The claim then follows. □

Lemma B.4. *Let $G = (V_1, V_2, E)$ be a bipartite graph and suppose that M is* any *semi-matching on G. Further, let $J = (V_1, V_2', E')$ be the subgraph of G restricted to only the nodes $V_2' \subseteq V_2$ on the right. Starting with $M' \subseteq M$, we can use algorithm $\mathcal{A}_{SM2}$ in [HLLT03] to obtain an optimal semi-matching $\mathcal{N}$ for the subgraph J. Let the nodes in V_1 be indexed such that*

$deg_M(1) \geq deg_M(2) \geq \ldots \geq deg_M(|V_1|)$ and indexed again such that $deg_{\mathcal{N}}(1) \geq deg_{\mathcal{N}}(2) \geq \ldots \geq deg_{\mathcal{N}}(|V_1|)$. Then for any $1 \leq s \leq |V_1|$, we have $\sum_{i=1}^{s} deg_{\mathcal{N}}(i) \leq \sum_{i=1}^{s} deg_M(i)$, i.e. the sum of the top s-degrees can only decrease as we go from M to $\mathcal{N}$.

Proof. Note that if we restrict M to just the nodes V_2', we get a feasible semi-matching M' on the subgraph J. Algorithm $\mathcal{A}_{SM2}$ proceeds by the iterated removal of cost-reducing paths. Note that when a cost-reducing path is removed, load is transferred from a node with larger degree (in the current semi-matching) to a node with strictly smaller degree. To see this, let $\hat{P} = (v_1^{(1)}, v_2^{(1)}, v_1^{(2)}, \ldots, v_1^{(d)})$ be a cost-reducing path (see Section 2.1 in [HLLT03]) in some semi-matching $\bar{M}$ on J. This means that $deg_{\bar{M}}(v_1^{(1)}) > deg_{\bar{M}}(v_1^{(d)}) + 1$. When we eliminate the cost-reducing path $\hat{P}$, the degree of $v_1^{(1)}$ decreases by 1 and that of $v_1^{(d)}$ increases by 1, but still the new degree of $v_1^{(d)}$ is strictly lower than the old degree of $v_1^{(1)}$. In other words, if $d_1^{\text{bef}} \geq d_2^{\text{bef}} \geq \ldots \geq d_{|V_1|}^{\text{bef}}$ and $d_1^{\text{aft}} \geq d_2^{\text{aft}} \geq \ldots \geq d_{|V_1|}^{\text{aft}}$ be the degree sequence (in the current semi-matching) before and after the removal of a cost-reducing path, then $\sum_{i=1}^{s} d_i^{\text{aft}} \leq \sum_{i=1}^{s} d_i^{\text{bef}}$ for any $1 \leq s \leq |V_1|$. Since this invariant is satisfied after every iteration of algorithm $\mathcal{A}_{SM2}$, it holds at the beginning and the end, and we have

$$\sum_{i=1}^{s} deg_{\mathcal{N}}(i) \leq \sum_{i=1}^{s} deg_{M'}(i) \tag{B.16}$$

Finally, observe that when we restrict M to only the set V_2', the sum of the top s-degrees can only *decrease*, i.e.

$$\sum_{i=1}^{s} deg_{M'}(i) \leq \sum_{i=1}^{s} deg_M(i) \tag{B.17}$$

Combining equations (B.16) and (B.17), the result follows. □

Lemma B.5. *Let $G = (V_1, V_2, E)$ be a bipartite graph and suppose that M is* any *semi-matching on G. Further, let $J = (V_1 \cup V_3, V_2, E')$ be the supergraph of G obtained by adding nodes V_3 on the left and edges $E' \setminus E$ to G. Starting with M, we can use algorithm $\mathcal{A}_{SM2}$ in [HLLT03] to obtain an optimal semi-matching $\mathcal{N}$ for the supergraph J. Let the nodes in V_1 be indexed such that $deg_M(1) \geq deg_M(2) \geq \ldots \geq deg_M(|V_1|)$ and the nodes in $V_1 \cup V_3$ indexed again such that $deg_{\mathcal{N}}(1) \geq deg_{\mathcal{N}}(2) \geq \ldots \geq deg_{\mathcal{N}}(|V_1 \cup V_3|)$. Then for any $1 \leq s \leq |V_1 \cup V_3|$, we have $\sum_{i=1}^{s} deg_{\mathcal{N}}(i) \leq \sum_{i=1}^{s} deg_M(i)$, i.e. the sum of the top s-degrees can only decrease as we go from M to $\mathcal{N}$.*

Proof. Note that M is a feasible semi-matching for the graph J, with all nodes in V_3 having degree 0. We can repeat the argument from the previous lemma to show that if $d_1^{\text{bef}} \geq d_2^{\text{bef}} \geq \ldots \geq d_{|V_1 \cup V_3|}^{\text{bef}}$ and $d_1^{\text{aft}} \geq d_2^{\text{aft}} \geq \ldots \geq d_{|V_1 \cup V_3|}^{\text{aft}}$ be the degree sequence (in the current semi-matching) before and after the removal of any cost-reducing path, then $\sum_{i=1}^{s} d_i^{\text{aft}} \leq \sum_{i=1}^{s} d_i^{\text{bef}}$ for any $1 \leq s \leq |V_1 \cup V_3|$. The result then follows. □

Notation for the proofs. Let $\mathcal{T}^+$ denote the set $\{t^+ : t \in \mathcal{T}\}$, and similarly $\mathcal{T}^-$ denote the set $\{t^- : t \in \mathcal{T}\}$, these are "task copies". Now partition the set of task copies $\mathcal{T}^+ \cup \mathcal{T}^-$

as $E \cup F$ such that for any task t, if the **true** label is $+1$, we put t^+ in E and t^- in F, otherwise, we put t^- in E and t^+ in F. Thus, E contains task copies with *true* labels while F contains task copies with *incorrect* labels. Let $F' \subseteq F$ denote the set of tasks for which honest workers provide incorrect responses, and denote the optimal semi-matching on the graph $\mathcal{B}_H^{cs}$ by $\mathcal{M}_H$. Without loss of generality, suppose that honest workers are indexed such that $d_1 \geq d_2 \geq \cdots \geq d_{|H|}$ where d_h denotes the degree of honest worker h in semi-matching $\mathcal{M}_H$.

B.5.1 Part 1: Adversary strategy that affects at least $\frac{1}{2}\sum_{i=1}^{k-1} d_i$ tasks

If $\varepsilon = 0$, there are no conflicts in the responses provided by honest workers, and therefore the bipartite graph $\mathcal{B}_H^{cs}$ contains just "true" task copies E on the right.

Given this, the adversaries *target* honest workers $\{1, 2, \ldots, k-1\}$: for each i, adversary a_i labels opposite to worker h_i (i.e. provides the incorrect response) on every task that h_i is mapped to in the semi-matching $\mathcal{M}_H$. Furthermore, the adversary uses its last identity a_k to provide the incorrect response on every task $t \in \mathcal{T}$ for which one of the first $k-1$ adversaries have not already labeled on. We argue that under the penalty-based label aggregation algorithm, this adversary strategy results in incorrect labels for at least $\frac{1}{2}\sum_{i=1}^{k-1} d_i$ tasks. To see this, first note that the conflict set $\mathcal{T}_{cs} = \mathcal{T}$. Further, the bipartite graph $\mathcal{B}^{cs}$ decomposes into two disjoint subgraphs: bipartite graph $\mathcal{B}^{cs}(E)$ between H and E and semi-matching $M(F)$ between A and F that represents the adversary labeling strategy (it is a semi-matching because there is exactly one adversary that labels each task). Since the bipartite graph $\mathcal{B}^{cs}$ decomposes into two disjoint bipartite graphs, computing the optimal semi-matching on $\mathcal{B}^{cs}$ is equivalent to separately computing optimal semi-matchings on $\mathcal{B}^{cs}(E)$ and $M(F)$. The argument above says that $\mathcal{B}^{cs}(E)$ is nothing but the graph $\mathcal{B}_H^{cs}$ defined in the theorem statement and therefore $\mathcal{M}_H$ is the optimal semi-matching on $\mathcal{B}^{cs}(E)$. Given that $M(F)$ is already a semi-matching by construction, the optimal semi-matching on $\mathcal{B}^{cs}$ is the disjoint union of $\mathcal{M}_H$ and $M(F)$. It is easy to see that in the resultant semi-matching, honest worker h_i and adversary a_i have the same degrees for $i = 1, 2, \ldots, k-1$. Hence, for every task mapped to honest worker h_i for $i = 1, 2, \ldots, k-1$ in the optimal semi-matching, the algorithm outputs a random label, and therefore outputs the correct label for at most half of these tasks. Thus, the above adversary strategy results in incorrect labels for at least $\frac{1}{2}\sum_{i=1}^{k-1} d_i$ tasks.

B.5.2 Part 2: Upper bound on number of affected tasks

To simplify the exposition, we assume in the arguments below that the optimal semi-matching in the hard-penalty algorithm is computed for the entire task set and not just the conflict set $\mathcal{T}_{cs}$. However, the bounds provided still hold as a result of Lemma B.4 above. Consequently, we abuse notation and use $\mathcal{B}^{cs}$ to denote the following bipartite graph in the remainder of the discussion: each worker w is represented by a node on the left, each task t is represented by at most two nodes on the right—t^+ and t^-—and we add an edge (w, t^+)

if worker w labels task t as $+1$ and edge (w, t^-) if w labels t as -1. Then, it is easy to see that the subgraph of $\mathcal{B}^{cs}$ restricted to just the honest workers H on the left and task copies $E \cup F'$ on the right, is nothing but the graph $\mathcal{B}^{cs}_H$ defined in the theorem statement. Consequently, $\mathcal{M}_H$ is a feasible semi-matching for the subgraph $\mathcal{B}^{cs}(E \cup F')$, which is obtained by restricting $\mathcal{B}^{cs}$ to nodes $E \cup F'$ on the right. Further, we can assume that the adversary labeling strategy is always a semi-matching, i.e. there is at most one adversary response for any task. If the adversary labeling strategy is not a semi-matching, they can replace it with an alternate strategy where they only label for tasks to which they will be mapped in the optimal semi-matching (the adversaries can compute this since they have knowledge of the honest workers' responses). The optimal semi-matching doesn't change (otherwise it contradicts the optimality of the original semi-matching) and hence neither does the number of affected tasks.

We first state the following important lemma:

Lemma B.6. *For any adversary labeling strategy, let $\mathcal{B}^{cs}(E \cup F')$ denote the bipartite graph $\mathcal{B}^{cs}$ restricted to all the workers $W = H \cup A$ on the left and, "true" task copies E as well as subset F' of the "incorrect" task copies F on the right. Let $\mathcal{M}$ be the optimal semi-matching on the bipartite graph $\mathcal{B}^{cs}$ and $\mathcal{M}(E \cup F') \subset \mathcal{M}$ be the semi-matching $\mathcal{M}$ restricted to only the task copies $E \cup F'$. Then, $\mathcal{M}(E \cup F')$ is an optimal semi-matching for the sub-graph $\mathcal{B}^{cs}(E \cup F')$.*

Proof. First observe that if $F' = F$, then the statement is trivially true. So we can assume $F' \subset F$. Suppose the statement is not true and let $\mathcal{N}(E \cup F')$ denote the optimal semi-matching on $\mathcal{B}^{cs}(E \cup F')$. We use $d_w(K)$ to denote the degree of worker w in a semi-matching K. Note that, $d_a(\mathcal{N}(E \cup F')) \leq d_a(\mathcal{M}(E \cup F')) \leq d_a(\mathcal{M})$ for all adversaries $a \in A$, where the first inequality follows from the fact that the adversary strategy is a semi-matching and the second inequality is true because $\mathcal{M}(E \cup F') \subset \mathcal{M}$. The adversaries who do not provide any responses for the task copies $E \cup F'$ will have degrees 0 in the semi-matchings $\mathcal{N}(E \cup F')$ and $\mathcal{M}(E \cup F')$ but the inequality is still satisfied. Now, since $\mathcal{N}(E \cup F')$ is an optimal semi-matching and $\mathcal{M}(E \cup F')$ is not, we have that

$$cost(\mathcal{N}(E \cup F')) < cost(\mathcal{M}(E \cup F')) \Rightarrow$$
$$\sum_{h \in H} d_h^2(\mathcal{N}(E \cup F')) + \sum_{a \in A} d_a^2(\mathcal{N}(E \cup F')) < \sum_{h \in H} d_h^2(\mathcal{M}(E \cup F')) + \sum_{a \in A} d_a^2(\mathcal{M}(E \cup F'))$$

Now, consider the semi-matching $\mathcal{N}$ on $\mathcal{B}^{cs}$ where we start with the semi-matching $\mathcal{N}(E \cup F')$ and then map the remaining task copies in $\mathcal{B}^{cs}$ (which belong to the set $F \setminus F'$) to the adversaries which they were assigned to in $\mathcal{M}$. We claim that $cost(\mathcal{N}) < cost(\mathcal{M})$ which is a

contradiction since $\mathcal{M}$ was assumed to be an optimal semi-matching on $\mathcal{B}^{cs}$. To see this:

$$
\begin{aligned}
&cost(\mathcal{M}) - cost(\mathcal{N}) \\
&= \sum_{h \in H} d_h^2(\mathcal{M}) + \sum_{a \in A} d_a^2(\mathcal{M}) - \left(\sum_{h \in H} d_h^2(\mathcal{N}) + \sum_{a \in A} d_a^2(\mathcal{N}) \right) \\
&= \sum_{h \in H} d_h^2(\mathcal{M}(E \cup F')) + \sum_{a \in A} \left(d_a(\mathcal{M}(E \cup F')) + \Delta_a \right)^2 \\
&- \left(\sum_{h \in H} d_h^2(\mathcal{N}(E \cup F')) + \sum_{a \in A} \left(d_a(\mathcal{N}(E \cup F')) + \Delta_a \right)^2 \right) \\
&\Big(\text{where } \Delta_a \stackrel{\text{def}}{=} d_a(\mathcal{M}) - d_a(\mathcal{M}(E \cup F')) \geq 0 \Big) \\
&= \left(\sum_{h \in H} d_h^2(\mathcal{M}(E \cup F')) + \sum_{a \in A} d_a^2(\mathcal{M}(E \cup F')) - \sum_{h \in H} d_h^2(\mathcal{N}(E \cup F')) - \sum_{a \in A} d_a^2(\mathcal{N}(E \cup F')) \right) \\
&+ 2 \sum_{a \in A} \left(d_a(\mathcal{M}(E \cup F')) - d_a(\mathcal{N}(E \cup F')) \right) * \Delta_a > 0 \\
&\big(\text{since } d_a(\mathcal{M}(E \cup F')) \geq d_a(\mathcal{N}(E \cup F')) \text{ as stated above} \big)
\end{aligned}
$$

Therefore, $\mathcal{M}(E \cup F')$ is an optimal semi-matching for the sub-graph $\mathcal{B}^{cs}(E \cup F')$. □

Part 2(a). Adversaries only provide incorrect responses or one adversary provides correct responses. We first prove part 2(a) in the theorem statement.

Adversaries only provide incorrect responses. Let us begin with the case when each adversary only provides incorrect responses.

Lemma B.7. *Suppose that $\varepsilon = 0$ and adversaries only provide incorrect responses. Let M be an* arbitrary *semi-matching on the bipartite graph $\mathcal{B}^{cs}$ and suppose that this semi-matching is used in the penalty-based label aggregation algorithm to compute the true labels of the tasks. Further, let $b_1 \geq b_2 \geq \cdots \geq b_{|H|}$ denote the degrees of the honest workers in this semi-matching where b_i is the degree of honest worker h_i. Then, the number of affected tasks is at most $\sum_{i=1}^{k} b_i$.*

Proof. It follows from the assumption that $\varepsilon = 0$ and that adversaries only provide incorrect responses, that there are no cross-edges between nodes H and F as well as A and E in the bipartite graph $\mathcal{B}^{cs}$. Thus, for any adversary labeling strategy, we can decompose $\mathcal{B}^{cs}$ into *disjoint* bipartite graphs $\mathcal{B}^{cs}(E)$ and $\mathcal{B}^{cs}(F)$, where $\mathcal{B}^{cs}(E)$ is the subgraph consisting of honest workers H and task copies E and $\mathcal{B}^{cs}(F)$ is the subgraph between the adversaries A and the task copies F. This further means that the semi-matching M is a disjoint union of semi-matchings on $\mathcal{B}^{cs}(E)$ and $\mathcal{B}^{cs}(F)$. Let the semi-matchings on the subgraphs be termed as $M(E)$ and $M(F)$ respectively. Further, let $T_{\text{aff}} \subseteq \mathcal{T}$ denote the set of tasks that are affected under this strategy of the adversaries and when the semi-matching M is used to compute the penalties of the workers in the penalty-based aggregation algorithm. We claim that $|T_{\text{aff}}| \leq \sum_{i=1}^{k} b_i$. To see this, for each adversary $a \in A$, let $H(a) \subset H$ denote the set

of honest workers who have "lost" to a i.e., for each worker $h \in H(a)$ there exists some task $t \in \mathcal{T}_{cs}$ such that h is mapped to the *true* copy of t in $M(E)$, a is mapped to the *incorrect* copy of t in $M(F)$, and the degree of h in $M(E)$ is greater than or equal to the degree of a in $M(F)$. Of course, $H(a)$ may be empty. Let $\bar{A}$ denote the set of adversaries $\{a \in A \colon H(a) \neq \emptyset\}$ and let $\bar{H}$ denote the set of honest workers $\bigcup_{a \in \bar{A}} H(a)$. Now define a bipartite graph between the nodes $\bar{A}$ and $\bar{H}$ with an edge between $a \in \bar{A}$ and $h \in \bar{H}$ if and only if $h \in H(a)$. This bipartite graph can be decomposed into degenerate and growth subgraphs by Lemma B.3 above. In the growth subgraph, by Hall's condition, we can find a perfect matching from adversaries to honest workers. Let (A_1, H_1) with $A_1 \subseteq \bar{A}$ and $H_1 = \mathrm{Img}(A_1)$ be the degenerate component. The number of tasks that adversaries in A_1 affect is bounded above by $\sum_{h \in \mathrm{Img}(A_1)} b_h$. Similarly, for $A_2 = \bar{A} \setminus A_1$, we can match each adversary to a *distinct* honest worker whose degree in $M(E)$ is greater than or equal to the degree of the adversary in $M(F)$. We can bound the number of affected tasks caused due to the adversaries in A_2 by the sum of their degrees, which in turn is bounded above by the sum of the degrees of honest workers that the adversaries are matched to. Let H_2 denote the set of honest workers matched to adversaries in the perfect matching. Thus, we have upper bounded the number of affected tasks by $\sum_{h \in H_1 \cup H_2} b_h$. It is easy to see that $|H_1 \cup H_2| \leq k$. Therefore, $\sum_{h \in H_1 \cup H_2} b_h \leq \sum_{i=1}^{k} b_i$. Therefore, the number of affected tasks $|T_{\text{aff}}|$ is at most $\sum_{i=1}^{k} b_i$. □

We now extend the above lemma to the case when $\varepsilon \neq 0$.

Lemma B.8. *Suppose that adversaries only provide incorrect responses. Let M be an* arbitrary *semi-matching on the bipartite graph $\mathcal{B}^{cs}$ and suppose that this semi-matching is used in the penalty-based aggregation algorithm to compute the true labels of the tasks. Further, let $b_1 \geq b_2 \geq \cdots \geq b_{|H|}$ denote the degrees of the honest workers in this semi-matching where b_i is the degree of honest worker h_i. Then, the number of affected tasks is at most $\sum_{i=1}^{k+\varepsilon \cdot |H|} b_i$.*

Proof. Since honest workers can make mistakes, there are cross-edges between H and F in the bipartite graph $\mathcal{B}^{cs}$. Observe that there are two kinds of affected tasks in this case: (i) tasks that adversaries "win" against honest workers, say $T_{\text{aff}}(A)$, similar to those in the previous lemma, and (ii) tasks that are affected when 2 honest workers are compared in the final step of the penalty-based aggregation algorithm, say $T_{\text{aff}}(H)$. We can repeat the argument from Lemma B.7 above to bound $|T_{\text{aff}}(A)|$ by the sum of the degrees of (some) k honest workers, say H_k, in semi-matching M. Further, we can bound $|T_{\text{aff}}(H)|$ by the sum of the degrees of honest workers who make mistakes, in the semi-matching M. By assumption, there are at most $\varepsilon \cdot |H|$ such workers. Now if any of these workers belong to H_k, then we have already accounted for their degree. Consequently, we have that $|T_{\text{aff}}(H)| + |T_{\text{aff}}(A)|$ is upper bounded by the sum of the degrees of the top $k + \varepsilon \cdot |H|$ honest workers in the semi-matching M, which establishes the result. □

Since the above lemmas are true for *any* choice of semi-matching M, they hold in particular for the optimal semi-matching on $\mathcal{B}^{cs}$. Therefore, it gives us an upper bound on the number

of affected tasks when the adversaries only provide incorrect responses.

One adversary provides correct responses. Next consider the case when there is exactly 1 adversary that provides correct responses (for some tasks) and all other adversaries only provide incorrect responses. Let $\mathcal{M}$ be the optimal semi-matching on the bipartite graph $\mathcal{B}^{cs}$ resulting from such an adversary strategy and let $\bar{a}$ denote the adversary who provides correct responses. Observe that we can repeat the argument from Lemma B.8 above to get an upper bound on the number of affected tasks that the adversaries "win" against honest workers as well as those that honest workers who make mistakes "win" against other honest workers. Let T_1 be the collection of such tasks. In the proof of Lemma B.7, there are two possible scenarios: either we obtain a perfect matching between the k adversaries and some k honest workers in which case we have accounted for all of the affected tasks that come from adversaries winning (against honest workers or $\bar{a}$). In the other scenario, when the degenerate component is non-empty, we have at most $k-1$ honest workers on the right and we bound the number of tasks that adversaries "win" by the sum of the degrees of these honest workers. Note however that we may be missing out on some of the affected tasks, namely those that the adversary $\bar{a}$ "loses" against *other adversaries* (the losses against honest workers who make mistakes are already accounted for). The tasks that we might be missing out on correspond exactly to the task copies in E that the adversary $\bar{a}$ is mapped to in the optimal semi-matching $\mathcal{M}$.

Next observe that in both scenarios above—all adversaries provide incorrect responses and exactly one adversary provides correct responses—we can upper bound the total number of affected tasks by the sum of the degrees of some $k + \varepsilon \cdot |H|$ workers in the optimal semi-matching $\mathcal{M}$ restricted to just the task copies $E \cup F'$ on the right (since honest workers provide responses only on these tasks), which we denote as $\mathcal{M}(E \cup F')$. Lemma B.6 tells us that $\mathcal{M}(E \cup F')$ is, in fact, the optimal semi-matching on the subgraph $\mathcal{B}^{cs}(E \cup F')$ between workers W and the task copies $E \cup F'$. In addition, Lemma B.5 tells us that this sum is at most $\sum_{i=1}^{k+\varepsilon\cdot|H|} d_i$ (by starting with $\mathcal{M}_H$ as a feasible semi-matching) and the bound follows.

Part 2(b). Adversaries can provide arbitrary responses. Consider the general case when all adversaries can provide arbitrary responses. First recall that Lemma B.8 was applicable to any semi-matching and in fact, we can use the argument even when adversaries provide correct responses. Formally, consider an arbitrary adversary strategy resulting in an optimal semi-matching $\mathcal{M}$ on $\mathcal{B}^{cs}$. Let $\mathcal{M}(E \cup F')$ denote the semi-matching M restricted to just the task copies $E \cup F'$. Suppose that the set of affected tasks T_{aff} under this adversary strategy is such that $T_{\text{aff}} = T_{\text{aff}}(A_1) \cup T_{\text{aff}}(H) \cup T_{\text{aff}}(A_2)$ where $T_{\text{aff}}(A_1)$ are the tasks that the adversaries "win" against honest workers, $T_{\text{aff}}(H)$ are the tasks that honest workers who make mistakes "win" (against other honest workers and/or adversaries) and $T_{\text{aff}}(A_2)$ are the tasks that are affected when 2 adversaries are compared against each other in the final step of the penalty-based aggregation algorithm. We can then utilize the argument in Lemma B.8 to bound $|T_{\text{aff}}(H)| + |T_{\text{aff}}(A_1)|$ by the sum of the degrees of (some) $k + \varepsilon \cdot |H|$ honest workers

in the optimal semi-matching $\mathcal{M}$ (the tasks affected when honest workers who make mistakes win against adversaries are also accounted). Further, we can bound $|T_{\text{aff}}(A_2)|$ by the sum of the degrees of the adversaries in the semi-matching $\mathcal{M}(E)$, which is the semi-matching $\mathcal{M}$ restricted to task copies E.

Let $A(H) \subseteq A$ denote the adversaries that have non-zero degrees in semi-matching $\mathcal{M}(E)$, i.e. they are mapped to some task copy in E in semi-matching $\mathcal{M}$. The above sequence of claims implies that we can bound the number of affected tasks $|T_{\text{aff}}|$ by the sum of the degrees of the top $s = k + \varepsilon \cdot |H| + |A(H)|$ workers in the semi-matching $\mathcal{M}(E \cup F')$, which is $\mathcal{M}$ restricted to the task copies $E \cup F'$ (because honest workers, by assumption, only provide responses on task copies $E \cup F'$ and the semi-matching $\mathcal{M}(E) \subseteq \mathcal{M}(E \cup F')$). Now, we claim that this itself is upper bounded by the sum of the degrees of the top s honest workers in the optimal semi-matching $\mathcal{M}_H$ on the bipartite graph $\mathcal{B}^{cs}_H$. To see this, start with $\mathcal{M}_H$ as a feasible semi-matching from workers W to task copies $E \cup F'$ (recall that it is feasible since we assume that each task has at least one correct response from honest workers). Then, Lemma B.5 tells us that the sum of the degrees of the top s workers in the optimal semi-matching on $\mathcal{B}^{cs}(E \cup F')$ is at most the sum of the degrees of the top s honest workers in $\mathcal{M}_H$. Further, Lemma B.6 tells us that the optimal semi-matching on the subgraph $\mathcal{B}^{cs}(E \cup F')$ is precisely the semi-matching $\mathcal{M}(E \cup F')$. This shows that we can bound the number of affected tasks by $\sum_{i=1}^{s} d_i$. Finally, note that $|A(H)| \leq k \Rightarrow s \leq 2k + \varepsilon \cdot |H|$ and hence, we can bound the total number of affected tasks by $\sum_{i=1}^{2k+\varepsilon\cdot|H|} d_i$.

B.5.3 Uniqueness of degree-sequence in optimal semi-matchings

In the arguments above, we have implicitly assumed some sort of uniqueness for the optimal semi-matching on any bipartite graph. Clearly its possible to have multiple optimal semi-matchings for a given bipartite graph. However, we prove below that the degree sequence of the vertices is unique across all optimal semi-matchings and hence our bounds still hold without ambiguity.

Lemma B.9. *Let M and M' be two optimal semi-matchings on a bipartite graph $G = (V_1 \cup V_2, E)$ with $|V_1| = n$ and let $d_1 \geq d_2 \cdots \geq d_n$ and $d'_1 \geq d'_2 \geq \cdots d'_n$ be the degree sequence for the V_1-vertices in M and M' respectively. Then, $d_i = d'_i \; \forall \; 1 \leq i \leq n$, or in other words, any two optimal semi-matchings have the same degree sequence.*

Proof. Let s be the smallest index such that $d_s \neq d'_s$, note that we must have $s < n$ since we have that $\sum_{j=1}^{n} d'_j = \sum_{j=1}^{n} d_j$. This means that we have $d_j = d'_j \; \forall j < s$. Without loss of generality, assume that $d'_s > d_s$. Now, $\exists \bar{p} \in \mathbb{N}$ such that $(d'_s)^{\bar{p}} > (d_s)^{\bar{p}} + \sum_{j=s+1}^{n} (d_j)^{\bar{p}}$ and since $d_j = d'_j \; \forall j < s$, we have that $\sum_{j=1}^{n} (d'_j)^{\bar{p}} \geq \sum_{j=1}^{l} (d'_j)^{\bar{p}} > \sum_{j=1}^{n} (d_j)^{\bar{p}}$. But, this is a contradiction since an optimal semi-matching minimizes the ℓ_p norm of the degree-vector of V_1-vertices for any $p \geq 1$ (Section 3.4 in [HLLT03]). Hence, we have that $d_i = d'_i \; \forall i$. □

B.5.4 Relation between lower bound L and optimal semi-matching degrees

We prove here the relationship between the lower bound L established in Theorem 2.10 and the honest worker degrees $d_1, d_2, \ldots, d_{|H|}$ in the optimal semi-matching on $\mathcal{B}_H^{cs}$.

Lemma B.10. *Suppose that honest workers are perfectly reliable and let $d_1 > d_2 > \cdots > d_{|H|}$ denote the degrees of the honest workers in the optimal semi-matching $\mathcal{M}_H$ on $\mathcal{B}_H^{cs}$. Then the lower bound L in Theorem 2.10 is such that $L \geq \sum_{i=1}^{k-1} d_i$.*

Proof. Let $T_1, T_2, \ldots, T_{k-1}$ denote the set of tasks that are mapped respectively to honest workers $h_1, h_2, \ldots, h_{k-1}$ in the optimal semi-matching $\mathcal{M}_H$ and $T := \bigcup_{j=1}^{k-1} T_j$. Now, we claim that for any $t \in T$, the only honest workers that provide responses for t are amongst $h_1, h_2, \ldots, h_k$. In other words, $\text{PreIm}(T) \subseteq \{h_1, h_2, \ldots, h_k\}$. Suppose not, so that there exists $h_i \in \text{PreIm}(T)$ such that $i > k$. This would contradict the fact that $\mathcal{M}_H$ is an optimal semi-matching. Specifically, Theorem 3.1 in [HLLT03] shows that a semi-matching M is optimal if and only if there is no *cost-reducing path* relative to M. A cost-reducing path $\hat{P} = (h^{(1)}, t^{(1)}, h^{(2)}, \ldots h^{(z)})$ for a semi-matching M on $\mathcal{B}_H^{cs}$ is an alternating sequence of honest workers and tasks such that $t^{(x)}$ is mapped to $h^{(x)}$ in the semi-matching M for all $1 \leq x \leq z-1$ and $deg_M(h^{(1)}) > deg_M(h^{(z)}) + 1$. Since $i > k$, we have that $d_s > d_i + 1$ for all $s \in \{1, 2, \ldots, k-1\}$, which introduces a cost-reducing path. Therefore, we have that $\text{PreIm}(T) \subseteq \{h_1, h_2, \ldots, h_k\}$. Then, it follows that

$$|T| = \left|\bigcup_{i=1}^{k-1} T_i\right| = \sum_{i=1}^{k-1} |T_i| = \sum_{i=1}^{k-1} d_i,$$

where we have used the property of a semi-matching that a given task is mapped to only one worker. Using the definition of the lower bound L, it follows that $L \geq |T| = \sum_{i=1}^{k-1} d_i$. □

B.6 Details of Numerical Analysis

EM algorithm for the two-coin model. We consider the EM algorithm proposed by [RY12]. The worker model they consider is as follows: for each worker w_i, her accuracy is modeled separately for positive and negative tasks (referred to as the "two-coin" model). For a task t_j with true label $+1$, the *sensitivity* (true positive rate) for worker w_i is defined as:

$$\alpha_i := \mathbb{P}[w_i(t_j) = +1 \,|\, y_j = +1]$$

Similarly, the *specificity* (1 - false positive rate) is defined as:

$$\beta_i := \mathbb{P}[w_i(t_j) = -1 \,|\, y_j = -1]$$

Let $\boldsymbol{\Theta} = [\{(\alpha_i, \beta_i) \,|\, i \in [n]\}, \gamma]$ denote the set of all parameters. Given the response matrix $\mathcal{L}$, the log-likelihood of the parameters $\boldsymbol{\Theta}$ can be written as:

$$\log \mathbb{P}[\mathcal{L} \mid \boldsymbol{\Theta}]$$
$$= \sum_{j=1}^{m} \log \left(\prod_{i \in W_j} \alpha_i^{\mathbf{1}[\mathcal{L}_{ij}=+1]} \cdot (1-\alpha_i)^{\mathbf{1}[\mathcal{L}_{ij}=-1]} \cdot \gamma + \prod_{i \in W_j} (1-\beta_i)^{\mathbf{1}[\mathcal{L}_{ij}=+1]} \cdot \beta_i^{\mathbf{1}[\mathcal{L}_{ij}=-1]} \cdot (1-\gamma) \right),$$

where recall that W_j is the set of workers assigned to task t_j. The MLE of the parameters can be computed by introducing the latent true label of each task, denoted by the vector $\boldsymbol{y} = [y_1, y_2, \ldots, y_m]$. The complete data log-likelihood can then be written as:

$$\log \mathbb{P}[\mathcal{L}, \boldsymbol{y} \mid \boldsymbol{\Theta}] = \sum_{j=1}^{m} \Big(y_j \cdot \log(a_j(\boldsymbol{\alpha}) \cdot \gamma) + (1-y_j) \log \left(b_j(\boldsymbol{\beta}) \cdot (1-\gamma) \right) \Big), \tag{B.18}$$

where

$$a_j(\boldsymbol{\alpha}) = \prod_{i \in W_j}^{n} \alpha_i^{\mathbf{1}[\mathcal{L}_{ij}=+1]} \cdot (1-\alpha_i)^{\mathbf{1}[\mathcal{L}_{ij}=-1]}$$
$$b_j(\boldsymbol{\beta}) = \prod_{i \in W_j}^{n} (1-\beta_i)^{\mathbf{1}[\mathcal{L}_{ij}=+1]} \cdot \beta_i^{\mathbf{1}[\mathcal{L}_{ij}=-1]}$$

Starting from an initial estimate $\boldsymbol{\Theta}^{(0)}$, each iteration $x \geq 1$ of the EM algorithm consists of two steps:

- **E-step:** Given the response matrix $\mathcal{L}$ and the current estimate of the model parameters $\boldsymbol{\Theta}^{(x-1)}$, the expected complete data log-likelihood (w.r.t to the conditional distribution $\boldsymbol{y} \mid \mathcal{L}; \boldsymbol{\Theta}^{(x-1)}$) is computed as

  $$\mathbb{E}_{\boldsymbol{y} \mid \mathcal{L}; \boldsymbol{\Theta}^{(x-1)}} \{ \log \mathbb{P}[\mathcal{L}, \boldsymbol{y} \mid \boldsymbol{\Theta}] \} = \sum_{j=1}^{m} \Big(\delta_j^{(x-1)} \log(a_j(\boldsymbol{\alpha}) \cdot \gamma) + (1-\delta_j^{(x-1)}) \log \left(b_j(\boldsymbol{\beta}) \cdot (1-\gamma) \right) \Big)$$

  where $\delta_j^{(x-1)} = \mathbb{P}[y_j = +1 \mid \mathcal{L}; \boldsymbol{\Theta}^{(x-1)}]$ is the current posterior estimate of the true label of task t_j being $+1$. Using Bayes theorem, we can compute

  $$\begin{aligned} \delta_j^{(x-1)} &\propto \mathbb{P}[\mathcal{L}_{1j}, \mathcal{L}_{2j}, \ldots, \mathcal{L}_{nj} \mid y_j = +1; \boldsymbol{\Theta}^{(x-1)}] \cdot \mathbb{P}[y_j = +1 \mid \boldsymbol{\Theta}^{(x-1)}] \\ &= \frac{a_j(\boldsymbol{\alpha}^{(x-1)}) \cdot \gamma^{(k)}}{a_j(\boldsymbol{\alpha}^{(x-1)}) \cdot \gamma^{(x-1)} + b_j(\boldsymbol{\beta}^{(x-1)}) \cdot (1-\gamma^{(x-1)})}. \end{aligned}$$

- **M-step:** Based on the current posterior estimates of the true labels $\delta_j^{(x-1)}$, the model parameters are updated by maximizing the expected complete data log-likelihood

$\mathbb{E}_{\boldsymbol{y}\,|\,\mathcal{L};\boldsymbol{\Theta}^{(x-1)}}\{\log \mathbb{P}[\mathcal{L}, \boldsymbol{y}\,|\,\boldsymbol{\Theta}]\}$, which can be shown to be a lower bound on the (incomplete) data log-likelihood $\log \mathbb{P}[\mathcal{L}\,|\,\boldsymbol{\Theta}]$. The prevalence of positive tasks γ is updated as:

$$\gamma^{(x)} = \frac{\sum_{j=1}^{m} \delta_j^{(x-1)}}{m}$$

Similarly, the parameters α_i, β_i are updated as:

$$\alpha_i^{(x)} = \frac{\sum_{j \in \mathcal{T}_i} \mathbf{1}[\mathcal{L}_{ij} = +1] \cdot \delta_j^{(x-1)}}{\sum_{j \in \mathcal{T}_i} \delta_j^{(x-1)}}$$
$$\beta_i^{(x)} = \frac{\sum_{j \in \mathcal{T}_i} \mathbf{1}[\mathcal{L}_{ij} = -1] \cdot (1 - \delta_j^{(x-1)})}{\sum_{j \in \mathcal{T}_i} (1 - \delta_j^{(x-1)})}$$

where recall that $\mathcal{T}_i$ denotes the set of tasks rated by worker w_i.

The above two steps are iterated until convergence of the log-likelihood $\log \mathbb{P}[\mathcal{L}\,|\,\boldsymbol{\Theta}]$. To initialize the EM algorithm, we use the majority estimate $\delta_j^{(0)} = (\sum_{i \in W_j} \mathbf{1}[\mathcal{L}_{ij} = +1])/\,|W_j|$.

KOS algorithms. We implemented the iterative algorithm presented in [KOS11] which we replicate below in our notation.

Algorithm 10 KOS LABEL AGGREGATION ALGORITHM

1: **Input:** response matrix $\mathcal{L}$, worker-task assignment graph $\mathcal{B} = (W \cup \mathcal{T}, E)$, max. number of iterations $x_{\max}$.

2: For all $(w_i, t_j) \in E$, initialize messages $z_{i \to j}^{(0)}$ with random $\mathbf{Z}_{ij} \sim \mathcal{N}(1, 1)$.

3: For $x = 1, 2, \ldots, x_{\max}$,

- For all $(w_i, t_j) \in E$, update $z_{j \to i}^{(x)} = \sum_{i' \neq i} \mathcal{L}_{ij} \cdot z_{i \to j}^{(x-1)}$
- For all $(w_i, t_j) \in E$, update $z_{i \to j}^{(x)} = \sum_{j' \neq j} \mathcal{L}_{ij} \cdot z_{j \to i}^{(x)}$

4: For all t_j, compute $z_j = \sum_{i=1}^{n} \mathcal{L}_{ij} \cdot z_{i \to j}^{(x_{\max})}$

5: **Output:** label for task t_j as $\hat{y}_j = \text{sign}(z_j)$

The algorithm above was specifically designed for random regular worker-task assignment graphs and we modified it in the following way for use in non-regular graphs:

Algorithm 11 KOS(NORM) LABEL AGGREGATION ALGORITHM

1: **Input:** response matrix $\mathcal{L}$, worker-task assignment graph $\mathcal{B} = (W \cup \mathcal{T}, E)$, max. number of iterations $x_{\max}$.

2: For all $(w_i, t_j) \in E$, initialize $z_{i\to j}^{(0)}$ with random $\mathbf{Z}_{ij} \sim \mathcal{N}(1, 1)$.

3: For $x = 1, 2, \ldots, x_{\max}$,

- For all $(w_i, t_j) \in E$, update $z_{j\to i}^{(x)} = \frac{1}{deg_{\mathcal{B}}(t_j)} \sum_{i' \neq i} \mathcal{L}_{ij} \cdot z_{i\to j}^{(x-1)}$
- For all $(w_i, t_j) \in E$, update $z_{i\to j}^{(x)} = \frac{1}{deg_{\mathcal{B}}(w_i)} \sum_{j' \neq j} \mathcal{L}_{ij} \cdot z_{j\to i}^{(x)}$

4: For all t_j, compute $z_j = \sum_{i=1}^{n} \mathcal{L}_{ij} \cdot z_{i\to j}^{(x_{\max})}$

5: **Output:** label for task t_j as $\hat{y}_j = \text{sign}(z_j)$

We chose $x_{\max} = 100$ in our experiments.

Degree bias for adversarial workers. We discuss here how we imposed the degree bias on adversaries in our simulation study. Given a worker-task assignment graph, let d_w denote the degree of worker w and $d_{\min}, d_{\text{avg}}, d_{\max}$ denote resp. the minimum, average and maximum worker degrees.

- *Adversaries are biased to have high degrees.* For each worker w, we compute the parameter $q_w = q \cdot \frac{d_{\max} - d_w}{d_{\max} - d_{\text{avg}}}$. Then, we sample the worker identities such that worker w is an adversary with probability $1 - q_w$. We claim that this process does not change the fraction of adversarial workers in the population (in expectation). To see that, the expected number of honest workers under this process is given by

$$\sum_w q_w = \frac{q}{d_{\max} - d_{\text{avg}}} \sum_w (d_{\max} - d_w) = q \cdot n.$$

  So the expected fraction of adversarial workers is given by $(n - q \cdot n)/n = 1 - q$. Also, it follows from the form of q_w that workers with higher degrees (close to $d_{\max}$) have smaller q_w values, which implies that they have a greater chance of being an adversary.

- *Adversaries are biased to have low degrees.* For each worker w, define $q_w = q \cdot \frac{d_w - d_{\min}}{d_{\text{avg}} - d_{\min}}$. Again, the worker identities are sampled such that each worker w is an adversary with probability $1 - q_w$. The expected number of honest workers under this process is given by

$$\sum_w q_w = \frac{q}{d_{\text{avg}} - d_{\min}} \sum_w (d_w - d_{\min}) = q \cdot n,$$

so that the fraction of adversarial workers remains at $1 - q$. In this case, lower the degree d_w, lower is the value of q_w and therefore higher the chance that worker w is an adversary.

Appendix C

Chapter 3 Proofs and Details of Numerical Experiments

C.1 Proof of Theorem 3.2

We first prove the rate for the squared loss, and then consider the negative log-likelihood loss.

C.1.1 Squared loss

For SQ loss, the convergence rate follows directly from existing results. For instance, Jaggi [Jag13] showed, for the optimization problem $\min_{\mathbf{x}\in\mathcal{D}} h(\mathbf{x})$ where $h(\cdot)$ is a differentiable convex function and $\mathcal{D}$ is a compact convex set, that the iterates of the fully corrective Frank-Wolfe variant (which is the one we consider) satisfy:

$$h(\mathbf{x}^{(k)}) - h(\mathbf{x}^*) \leq \frac{2 \cdot C_h}{k+2} \tag{C.1}$$

for all $k \geq 1$. Here C_h is the *curvature constant*—a measure of the "non-linearity"—of the function $h(\cdot)$ over the domain $\mathcal{D}$, defined as:

$$C_h := \sup_{\substack{\mathbf{x},\mathbf{s}\in\mathcal{D} \\ \gamma\in[0,1] \\ \mathbf{r}=\mathbf{x}+\gamma(\mathbf{s}-\mathbf{x})}} \frac{2}{\gamma^2} \cdot \Big(h(\mathbf{r}) - h(\mathbf{x}) - \langle \mathbf{r}-\mathbf{x}, \nabla h(\mathbf{x}) \rangle \Big).$$

Since $h(\cdot)$ is convex, the curvature constant $C_h \geq 0$. In addition, if the function $h(\cdot)$ is twice differentiable, then it can be shown [Jag11, Equation 2.12] that

$$C_h \leq \sup_{\substack{\mathbf{x},\mathbf{s}\in\mathcal{D} \\ \mathbf{r}\in[\mathbf{x},\mathbf{s}]\subseteq\mathcal{D}}} (\mathbf{s}-\mathbf{x})^\top \nabla^2 h(\mathbf{r})(\mathbf{s}-\mathbf{x}),$$

where $[\mathbf{x}, \mathbf{s}]$ is the line-segment joining $\mathbf{x}$ and $\mathbf{s}$—since $\mathcal{D}$ is convex, it lies within $\mathcal{D}$.

In our case, the convex objective $h = \mathsf{SQ}$ and the domain $\mathcal{D} = \text{conv}(\overline{\mathcal{P}})$. Further, the hessian $\nabla^2\mathsf{SQ}(\cdot)$ is a diagonal matrix with entry corresponding to product j in offer-set S_t as $(\nabla^2\mathsf{SQ}(\mathbf{r}))_{jt} = N_t/N$. Then, consider the following:

$$
\begin{aligned}
C_{\mathsf{SQ}} &\leq \sup_{\substack{\mathbf{x},\mathbf{s}\in\mathcal{D} \\ \mathbf{r}\in[\mathbf{x},\mathbf{s}]\subseteq\mathcal{D}}} (\mathbf{s}-\mathbf{x})^\top \nabla^2\mathsf{SQ}(\mathbf{r})(\mathbf{s}-\mathbf{x}) \\
&= \sup_{\mathbf{x},\mathbf{s}\in\mathcal{D}} \sum_{t=1}^{T} \sum_{j\in S_t} (s_{jt}-x_{jt})^2 \cdot \frac{N_t}{N} \\
&= \frac{1}{N} \cdot \sup_{\mathbf{x},\mathbf{s}\in\mathcal{D}} \sum_{t=1}^{T} N_t \cdot \sum_{j\in S_t} (s_{jt}-x_{jt})^2 \\
&\leq \frac{1}{N} \cdot \sup_{\mathbf{x},\mathbf{s}\in\mathcal{D}} \sum_{t=1}^{T} N_t \cdot \sum_{j\in S_t} |x_{jt}-s_{jt}| \qquad (\text{since } |x_{jt}-s_{jt}| \leq 1) \\
&\leq \frac{1}{N} \cdot \sup_{\mathbf{x},\mathbf{s}\in\mathcal{D}} \sum_{t=1}^{T} N_t \cdot \sum_{j\in S_t} (|x_{jt}|+|s_{jt}|) \qquad (\text{using triangle inequality}) \\
&= \frac{1}{N} \cdot \sup_{\mathbf{x},\mathbf{s}\in\mathcal{D}} \sum_{t=1}^{T} N_t \cdot (1+1) \qquad (\text{since choice probs. sum to 1 in each offer-set}) \\
&= \frac{1}{N} \cdot 2 \cdot N = 2
\end{aligned}
$$

The result then follows by plugging in $C_{\mathsf{SQ}} \leq 2$ in equation (C.1) above.

C.1.2 Negative log-likelihood loss

We refer to the domain $\text{conv}(\overline{\mathcal{P}})$ as $\mathcal{D}$ below for succinct notation. We establish the convergence rate assuming that $y_{jt} > 0$ for all $j \in S_t$ and all offer-sets S_t but the arguments below can be extended in a straight-forward manner to the case when some of the y_{jt}'s are zero by dropping terms for those product, offer-set pairs when defining the NLL loss objective (since they do not contribute anyway to the loss objective), and only maintaining the choice probabilities for the remaining product, offer-set pairs.

The proof proceeds in the following steps:

1. We show that there exists $\eta > 0$ such that the iterates $\mathbf{g}^{(k)}$ in Algorithm 8 satisfy $g_{jt}^{(k)} \geq \eta$ for all $j \in S_t$ and all $1 \leq t \leq T$, and all $k \geq 0$. The idea is that if any of the iterates get too close to 0, then the objective value $\mathsf{NLL}(\mathbf{g}^{(k)})$ will exceed the starting objective value $\mathsf{NLL}(\mathbf{g}^{(0)})$ which is a contradiction since the fully corrective variant of the Frank-Wolfe algorithm produces a decreasing objective value in each iteration.

2. Utilizing the lower bound computed in Step 1, we adapt arguments from existing work [GM86] to show that Algorithm 8 achieves $O(1/k)$ convergence to the optimal solution.

We first prove Step 2 assuming the existence of a lower bound η and then compute a tight value for η.

Step 2: convergence rate. In particular, we establish the following lemma:

Lemma C.1. *Suppose there exists $\eta > 0$ such that the iterates $\mathbf{g}^{(k)}$ in Algorithm 8 satisfy $\mathrm{g}_{jt}^{(k)} \geq \eta$ for all $j \in S_t$ and all $1 \leq t \leq T$, and all $k \geq 0$. Then, there exists index K and constant δ such that*

$$\mathsf{NLL}(\mathbf{g}^{(k)}) - \mathsf{NLL}(\mathbf{g}^*) \leq \frac{4}{\eta^2 \cdot (k+\delta)} \quad \forall\, k \geq K.$$

Proof. Define $\widetilde{\mathcal{D}} = \{\mathbf{g} \in \mathcal{D} \mid \mathrm{g}_{jt} \geq \frac{\eta}{\sqrt{2}} \;\; \forall\, j \in S_t \;\; \forall 1 \leq t \leq T\}$. At any iteration $k \geq 1$, let $\mathbf{d}^{(k)} := \boldsymbol{f}^{(k)} - \mathbf{g}^{(k-1)}$ where recall that $\boldsymbol{f}^{(k)} \in \arg\min_{\boldsymbol{f} \in \overline{\mathcal{P}}} \left\langle \nabla \mathsf{loss}(\mathbf{g}^{(k-1)}), \boldsymbol{f} \right\rangle$. Now, observe that for any $k \geq 1$:

$$\begin{aligned}
\mathsf{NLL}(\mathbf{g}^*) - \mathsf{NLL}(\mathbf{g}^{(k-1)}) &\geq \left\langle \mathbf{g}^* - \mathbf{g}^{(k-1)}, \nabla \mathsf{NLL}(\mathbf{g}^{(k-1)}) \right\rangle \quad \text{(since } \mathsf{NLL}(\cdot) \text{ is convex)} \\
&\geq \left\langle \boldsymbol{f}^{(k)} - \mathbf{g}^{(k-1)}, \nabla \mathsf{NLL}(\mathbf{g}^{(k-1)}) \right\rangle \quad \text{(using definition of } \boldsymbol{f}^{(k)}) \quad \text{(C.2)} \\
&= \left\langle \mathbf{d}^{(k)}, \nabla \mathsf{NLL}(\mathbf{g}^{(k-1)}) \right\rangle
\end{aligned}$$

Next, consider a step size $\gamma \in [0, 1]$ such that $\mathbf{g}^{(k-1)} + \gamma \cdot \mathbf{d}^{(k)} \in \widetilde{\mathcal{D}}$. Using second-order Taylor series approximation of $\mathsf{NLL}(\cdot)$ around the point $\mathbf{g}^{(k-1)}$, we have:

$$\mathsf{NLL}(\mathbf{g}^{(k-1)} + \gamma \cdot \mathbf{d}^{(k)}) = \mathsf{NLL}(\mathbf{g}^{(k-1)}) + \gamma \left\langle \mathbf{d}^{(k)}, \nabla \mathsf{loss}(\mathbf{g}^{(k-1)}) \right\rangle + \frac{\gamma^2}{2} \mathbf{d}^{(k)\top} \nabla^2 \mathsf{NLL}(\mathbf{r}_k) \mathbf{d}^{(k)},$$

where $\mathbf{r}_k$ lies on the line segment $[\mathbf{g}^{(k-1)}, \mathbf{g}^{(k-1)} + \gamma \cdot \mathbf{d}^{(k)}]$. Since $\mathbf{g}^{(k-1)} \in \widetilde{\mathcal{D}}$ and $\mathbf{g}^{(k-1)} + \gamma \cdot \mathbf{d}^{(k)} \in \widetilde{\mathcal{D}}$, it implies $\mathbf{r}_k \in \widetilde{\mathcal{D}}$ and consequently $r_{k,jt} \geq \frac{\eta}{\sqrt{2}}$ for all $j \in S_t$ and all $1 \leq t \leq T$. Then, consider the following:

$$\begin{aligned}
\mathsf{NLL}(\mathbf{g}^{(k-1)} + \gamma \cdot \mathbf{d}^{(k)}) &= \mathsf{NLL}(\mathbf{g}^{(k-1)}) + \gamma \left\langle \mathbf{d}^{(k)}, \nabla \mathsf{NLL}(\mathbf{g}^{(k-1)}) \right\rangle + \frac{\gamma^2}{2} \mathbf{d}^{(k)\top} \nabla^2 \mathsf{NLL}(\mathbf{r}_k) \mathbf{d}^{(k)} \\
&= \mathsf{NLL}(\mathbf{g}^{(k-1)}) + \gamma \left\langle \mathbf{d}^{(k)}, \nabla \mathsf{NLL}(\mathbf{g}^{(k-1)}) \right\rangle + \frac{\gamma^2}{2 \cdot N} \sum_{t=1}^{T} \sum_{j \in S_t} \frac{N_{jt} \cdot \mathrm{d}_{jt}^{(k)2}}{r_{k,jt}^2} \\
&\leq \mathsf{NLL}(\mathbf{g}^{(k-1)}) + \gamma \left\langle \mathbf{d}^{(k)}, \nabla \mathsf{NLL}(\mathbf{g}^{(k-1)}) \right\rangle + \frac{\gamma^2}{\eta^2 \cdot N} \sum_{t=1}^{T} \sum_{j \in S_t} N_{jt} \\
&\quad \left(\text{since } \left|\mathrm{d}_{jt}^{(k)}\right| \leq 1 \text{ and } r_{k,jt} \geq \frac{\eta}{\sqrt{2}} \quad \forall j \in S_t \;\; \forall 1 \leq t \leq T\right) \\
&\leq \mathsf{NLL}(\mathbf{g}^{(k-1)}) - \gamma \cdot \left(\mathsf{NLL}(\mathbf{g}^{(k-1)}) - \mathsf{NLL}(\mathbf{g}^*)\right) + \frac{\gamma^2}{\eta^2} \\
&\quad \{\text{using equation (C.2)}\}
\end{aligned}$$

Denoting $\mathsf{gap}(\mathbf{g}) = \mathsf{loss}(\mathbf{g}) - \mathsf{loss}(\mathbf{g}^*)$ as the optimality gap, and plugging it above we get

$$\mathsf{NLL}(\mathbf{g}^{(k-1)} + \gamma \cdot \mathbf{d}^{(k)}) \leq \mathsf{NLL}(\mathbf{g}^{(k-1)}) - \gamma \cdot \mathsf{gap}(\mathbf{g}^{(k-1)}) + \frac{\gamma^2}{\eta^2} \tag{C.3}$$

Now, choose $\gamma^* = \frac{\eta^2 \cdot \mathsf{gap}(\mathbf{g}^{(k-1)})}{2}$ and observe that γ^* minimizes the RHS of equation (C.3). With this observation, we establish a sequence of claims that will lead us to the result.

Claim 1: $\mathbf{g}^{(k-1)} + \gamma^* \cdot \mathbf{d}^{(k)} \in \widetilde{\mathcal{D}}$.
This means that for any $1 \leq t \leq T$ and any $j \in S_t$, we need to show:

$$\begin{aligned}
&\mathrm{g}_{jt}^{(k-1)} + \frac{\mathrm{d}_{jt}^{(k)} \cdot \eta^2}{2} \cdot \mathsf{gap}(\mathbf{g}^{(k-1)}) \geq \frac{\eta}{\sqrt{2}} \\
&\iff -\mathrm{d}_{jt}^{(k)} \cdot \mathsf{gap}(\mathbf{g}^{(k-1)}) \leq 2 \cdot \left(\frac{\mathrm{g}_{jt}^{(k-1)} - \frac{\eta}{\sqrt{2}}}{\eta^2} \right) \\
&\iff (\mathrm{g}_{jt}^{(k-1)} - f_{jt}^{(k)}) \cdot \mathsf{gap}(\mathbf{g}^{(k-1)}) \leq 2 \cdot \left(\frac{\mathrm{g}_{jt}^{(k-1)} - \frac{\eta}{\sqrt{2}}}{\eta^2} \right) \\
&\impliedby \mathrm{g}_{jt}^{(k-1)} \cdot \mathsf{gap}(\mathbf{g}^{(k-1)}) \leq 2 \cdot \left(\frac{\mathrm{g}_{jt}^{(k-1)} - \frac{\eta}{\sqrt{2}}}{\eta^2} \right) \qquad (\text{since } \mathsf{gap}(\mathbf{g}^{(k-1)}) \geq 0) \\
&\iff \frac{1}{N} \sum_{t=1}^{T} \sum_{\ell \in S_t} N_{\ell t} \log \frac{\mathrm{g}_{\ell t}^*}{\mathrm{g}_{\ell t}^{(k-1)}} \leq \frac{2}{\eta^2} - \frac{\sqrt{2}}{\eta \cdot \mathrm{g}_{jt}^{(k-1)}} \\
&\impliedby \frac{1}{N} \sum_{t=1}^{T} \sum_{\ell \in S_t} N_{\ell t} \cdot \left(\frac{\mathrm{g}_{\ell t}^*}{\mathrm{g}_{\ell t}^{(k-1)}} - 1 \right) \leq \frac{2}{\eta^2} - \frac{\sqrt{2}}{\eta \cdot \mathrm{g}_{jt}^{(k-1)}} \quad (\text{since } \log z \leq z - 1 \ \ \forall\, z > 0) \\
&\iff \frac{1}{N} \sum_{t=1}^{T} \sum_{\ell \in S_t} N_{\ell t} \cdot \left(\frac{\mathrm{g}_{\ell t}^*}{\mathrm{g}_{\ell t}^{(k-1)}} \right) \leq 1 + \frac{2}{\eta^2} - \frac{\sqrt{2}}{\eta \cdot \mathrm{g}_{jt}^{(k-1)}} \\
&\impliedby \frac{1}{N} \sum_{t=1}^{T} \sum_{\ell \in S_t} N_{\ell t} \cdot \left(\frac{1}{\eta} \right) \leq 1 + \frac{2}{\eta^2} - \frac{\sqrt{2}}{\eta \cdot \mathrm{g}_{jt}^{(k-1)}} \\
&(\text{since } 0 \leq \mathrm{g}_{\ell t}^* \leq 1 \text{ and } \mathrm{g}_{\ell t}^{(k-1)} \geq \eta \quad \forall \ell \in S_t \ \forall\, 1 \leq t \leq T) \\
&\impliedby \frac{1}{\eta} \leq 1 + \frac{2}{\eta^2} - \frac{\sqrt{2}}{\eta \cdot \eta} \quad (\text{since } \mathrm{g}_{jt}^{(k-1)} \geq \eta) \\
&\iff 0 \leq \eta^2 - \eta + 2 - \sqrt{2}
\end{aligned}$$

The final statement is true for any $\eta > 0$ and therefore the claim follows. In addition, it is easy to see that $\mathbf{g}^{(k-1)} + \gamma \cdot \mathbf{d}^{(k)} \in \widetilde{\mathcal{D}}$ for all $\gamma \in [0, m_k]$ where $m_k \stackrel{\text{def}}{=} \min\left(1, \frac{\eta^2 \cdot \mathsf{gap}(\mathbf{g}^{(k-1)})}{2}\right)$.

Claim 2:

$$\mathsf{gap}(\mathbf{g}^{(k)}) \leq \mathsf{gap}(\mathbf{g}^{(k-1)}) \cdot \left(1 - \frac{m_k}{2}\right) \quad \forall\, k \geq 1. \tag{C.4}$$

Consider the following:

$$
\begin{aligned}
\mathsf{NLL}(\mathbf{g}^{(k)}) &\leq \min_{\gamma \in [0,1]} \mathsf{NLL}(\mathbf{g}^{(k-1)} + \gamma \cdot \mathbf{d}^{(k)}) \\
&\quad \text{(since FCFW guarantees as much progress as line-search FW)} \\
&\leq \min_{\gamma \in [0, m_k]} \mathsf{NLL}(\mathbf{g}^{(k-1)} + \gamma \cdot \mathbf{d}^{(k)}) \\
&\leq \min_{\gamma \in [0, m_k]} \mathsf{NLL}(\mathbf{g}^{(k-1)}) - \gamma \cdot \mathsf{gap}(\mathbf{g}^{(k-1)}) + \frac{\gamma^2}{\eta^2} \\
&\quad \Big(\text{since } \mathbf{g}^{(k-1)} + \gamma \cdot \mathbf{d}^{(k)} \in \widetilde{\mathcal{D}} \ \ \forall\, \gamma \in [0, m_k] \text{ and using equation (C.3)}\Big) \\
&= \mathsf{NLL}(\mathbf{g}^{(k-1)}) - m_k \cdot \mathsf{gap}(\mathbf{g}^{(k-1)}) + \frac{m_k^2}{\eta^2} \\
&\quad \text{(by choice of } m_k) \\
&\leq \mathsf{NLL}(\mathbf{g}^{(k-1)}) - m_k \cdot \mathsf{gap}(\mathbf{g}^{(k-1)}) + \frac{m_k \cdot \eta^2 \cdot \mathsf{gap}(\mathbf{g}^{(k-1)})}{2 \cdot \eta^2} \\
&\quad \left(\text{since } m_k \leq \frac{\eta^2 \cdot \mathsf{gap}(\mathbf{g}^{(k-1)})}{2}\right) \\
&= \mathsf{NLL}(\mathbf{g}^{(k-1)}) - \frac{m_k}{2} \cdot \mathsf{gap}(\mathbf{g}^{(k-1)})
\end{aligned}
$$

Then subtracting $\mathsf{NLL}(\mathbf{g}^*)$ from both sides and using the definition of m_k, the claim follows.

Claim 3: $\lim_{k\to\infty} m_k = 0$.

Since both sequences $\{\mathsf{NLL}(\mathbf{g}^{(k)})\}_k$ and $\{m_k\}_k$ are non-increasing and bounded below by zero, it follows by taking limits on both sides of equation (C.4) that:

$$
\begin{aligned}
&\lim_{k\to\infty} \mathsf{gap}(\mathbf{g}^{(k)}) \leq \lim_{k\to\infty} \mathsf{gap}(\mathbf{g}^{(k-1)}) \cdot \left(1 - \frac{1}{2} \lim_{k\to\infty} m_k\right) \\
&\implies \lim_{k\to\infty} \mathsf{gap}(\mathbf{g}^{(k)}) \cdot \lim_{k\to\infty} m_k \leq 0 \\
&\implies \lim_{k\to\infty} \mathsf{gap}(\mathbf{g}^{(k)}) = 0 \quad \text{or} \quad \lim_{k\to\infty} m_k = 0 \quad \big(\text{since } m_k \geq 0 \text{ and } \mathsf{gap}(\mathbf{g}^{(k)}) \geq 0 \ \ \forall k\big) \\
&\implies \lim_{k\to\infty} m_k = 0 \quad \text{(using definition of } m_k)
\end{aligned}
$$

Given the above claims, let K be the smallest iteration number such that $m_K \leq 1$. Then

from equation (C.4) it follows that

$$
\begin{aligned}
& \mathsf{gap}(\mathbf{g}^{(k)}) \leq \mathsf{gap}(\mathbf{g}^{(k-1)}) \cdot \left(1 - \frac{m_k}{2}\right) \quad \forall\, k \geq K \\
\iff & \frac{m_{k+1}}{2} \leq \frac{m_k}{2} \cdot \left(1 - \frac{m_k}{2}\right) \quad \forall\, k \geq K \\
\iff & \frac{2}{m_{k+1}} \geq \frac{2}{m_k} \cdot \frac{2}{2 - m_k} \quad \forall\, k \geq K \\
\iff & \frac{2}{m_{k+1}} \geq \frac{2}{m_k} \cdot \left(1 + \frac{m_k}{2 - m_k}\right) \quad \forall\, k \geq K \\
\implies & \frac{2}{m_{k+1}} \geq \frac{2}{m_k} + 1 \quad \forall\, k \geq K \quad (\text{since } m_k \leq 1 \ \ \forall \ \ k \geq K) \\
\implies & \frac{2}{m_k} \geq k - K + \frac{2}{m_K} \quad \forall\, k \geq K \\
\iff & m_k \leq \frac{2}{k + \tilde{\delta}} \quad \forall\, k \geq K \quad \left(\text{where } \tilde{\delta} = \frac{2}{m_K} - K\right) \\
\iff & \mathsf{gap}(\mathbf{g}^{(k-1)}) \leq \frac{4}{\eta^2 \cdot (k + \tilde{\delta})} \quad \forall\, k \geq K \\
\iff & \mathsf{gap}(\mathbf{g}^{(k)}) \leq \frac{4}{\eta^2 \cdot (k + \delta)} \quad \forall\, k \geq K \left(\text{where } \delta = 1 + \frac{2}{m_K} - K\right)
\end{aligned}
$$

from which the result follows. □

Step 1: lower bound for iterates. Lemma C.1 establishes $1/k$ convergence rate of Algorithm 8 given *any* lower bound $\eta > 0$ for the iterates $\mathbf{g}^{(k)}$. We now come up with a tight lower bound— $\xi_{\min}$ in the main text, based on the initialization, i.e. we show that

$$
\mathrm{g}_{jt}^{(k)} \geq \xi_{\min} \ \ \forall\, j \in S_t \ \ \forall\, 1 \leq t \leq T \text{ and } \forall\, k \geq 0
$$

For any vector $\mathbf{x} = (x_1, x_2, \ldots, x_m)$, denote $x_{\min} \stackrel{\text{def}}{=} \min_{i=1,2,\ldots,m} x_i$. Then, for any $\xi \in (0, 1]$, consider the following optimization problem:

$$
G(\xi) \equiv \min_{\mathbf{x} \in \mathbb{R}^M} \mathsf{NLL}(\mathbf{x}) \quad \text{s.t. } x_{jt} \geq 0 \ \ \forall\, j \in S_t; \ \sum_{j \in S_t} x_{jt} = 1; \quad \forall 1 \leq t \leq T \text{ and } x_{\min} \leq \xi \tag{C.5}
$$

We first come up with a closed-form expression for $G(\xi)$. For each $1 \leq t \leq T$ and every $i \in S_t$, define the following optimization problem:

$$
G_{i,t}(\xi) \equiv \min_{\mathbf{x} \in \mathbb{R}^M} \mathsf{NLL}(\mathbf{x}) \quad \text{s.t. } x_{jt'} \geq 0 \ \ \forall\, j \in S_{t'}; \ \sum_{j \in S_{t'}} x_{jt'} = 1; \quad \forall 1 \leq t' \leq T \text{ and } x_{it} \leq \xi \tag{C.6}
$$

Claim 1:

$$
G(\xi) = \min_{1 \leq t \leq T} \min_{i \in S_t} G_{i,t}(\xi) \ \text{ for all } \xi \in (0, 1]
$$

It is easy to see that $\min_{1\le i\le t}\min_{i\in S_t} G_{i,t}(\xi) \le G(\xi)$ for any $\xi \in (0,1]$ since the optimal solution for problem (C.5) is feasible for some product, offer-set pair (i,t).

For the other direction, suppose $\min_{1\le t\le T}\min_{i\in S_t} G_{i,t}(\xi) = G_{j^*,t^*}(\xi)$ for some $j^* \in S_{t^*}$. Let $\mathbf{x}^*$ denote the optimal solution for $G_{j^*,t^*}(\xi)$, so that $x^*_{j^*,t^*} \le \xi$. This also means that $x^*_{\min} \le \xi$ and consequently $\mathbf{x}^*$ is a feasible solution for problem (C.5). Therefore, $G(\xi) \le G_{j^*,t^*}(\xi) = \min_{1\le t\le T}\min_{i\in S_t} G_{i,t}(\xi)$ and the claim then follows.

Claim 2:

$$G_{i,t}(\xi) = \frac{1}{N}\cdot\left(N_t \cdot D_{KL}(y_{it}\|\xi) + \sum_{t'=1}^{T} N_{t'}\cdot H(\boldsymbol{y}_{t'})\right) \quad \forall\, \xi \in (0, y_{it}].$$

This follows because the optimal solution, say $\mathbf{x}^*$, to problem (C.6) is of the form: (1) $x^*_{jt'} = y_{jt'}$ for all $j \in S_{t'}$, $\forall t' \ne t$ (2) $x^*_{jt} = \frac{y_{jt}}{1-y_{it}}\cdot(1-\xi)$ for all $j \in S_t \setminus \{i\}$ and (3) $x^*_{it} = \xi$.

This can be verified by solving the KKT conditions for problem (C.6). In particular, letting $\lambda_{t'}$ denote the dual variable for the constraint $\sum_{j\in S_{t'}} x_{jt'} = 1$, and μ denote the dual variable for the constraint $x_{it} \le \xi$, the KKT conditions are given by:

$$\frac{N_{jt'}}{x_{jt'}\cdot N} = \lambda_{t'} \;\; \forall j \in S_{t'} \text{ and } \forall t' \ne t; \quad \frac{N_{jt}}{x_{jt}\cdot N} = \lambda_t \;\; \forall j \in S_t \setminus \{i\}; \quad \frac{N_{it}}{x_{it}\cdot N} = \lambda_t + \mu \quad \text{(Stationarity)}$$

$$\mu\cdot(x_{it}-\xi) = 0 \quad \text{(Complementary slackness)}$$

$$\mu \ge 0 \quad \text{(Dual feasibility)}$$

$$\sum_{j\in S_{t'}} x_{jt'} = 1; \;\; x_{jt'} \ge 0 \;\; \forall j \in S_{t'} \;\forall\; 1 \le t' \le T \quad \text{(Primal feasibility)}$$

Solving these equations gives the optimal solution mentioned above. Plugging the optimal solution $\mathbf{x}^*$ into the $\mathsf{NLL}(\cdot)$ loss objective, we obtain

$$\begin{aligned}
G_{i,t}(\xi) &= \frac{1}{N}\cdot\left(-N_{it}\log\xi - \sum_{j\in S_t\setminus\{i\}} N_{jt}\log\frac{y_{jt}\cdot(1-\xi)}{1-y_{it}} - \sum_{t'\ne t}\sum_{j\in S_{t'}} N_{jt'}\log\left(y_{jt'}\right)\right)\\
&= \frac{1}{N}\cdot\left(-N_{it}\log\xi - \sum_{j\in S_t\setminus\{i\}} N_{jt}\log\frac{(1-\xi)}{1-y_{it}} - \sum_{j\in S_t\setminus\{i\}} N_{jt}\log y_{jt} + \sum_{t'\ne t} N_{t'}\cdot H(\boldsymbol{y}_{t'})\right)\\
&= \frac{1}{N}\cdot\left(-N_{it}\log\xi - (N_t - N_{it})\log\frac{1-\xi}{1-y_{it}} + N_{it}\log y_{it} + N_t\cdot H(\boldsymbol{y}_t) + \sum_{t'\ne t} N_{t'}\cdot H(\boldsymbol{y}_{t'})\right)\\
&= \frac{N_t}{N}\cdot\left(y_{it}\log\frac{y_{it}}{\xi} + (1-y_{it})\log\frac{1-y_{it}}{1-\xi}\right) + \frac{1}{N}\sum_{t'=1}^{T} N_{t'}\cdot H(\boldsymbol{y}_{t'})\\
&= \frac{1}{N}\cdot\left(N_t\cdot D_{KL}(y_{it}\|\xi) + \sum_{t'=1}^{T} N_{t'}\cdot H(\boldsymbol{y}_{t'})\right)
\end{aligned}$$

where $D_{KL}(y_{it}\|\xi)$ is the relative entropy between y_{it} and ξ, and $H(\boldsymbol{y}_{t'})$ is the entropy of vector $\boldsymbol{y}_{t'}$.

Claim 3: For each $1 \leq t \leq T$ and any $\xi \in (0, y_{t,\min}]$, it follows that

$$\min_{i \in S_t} G_{i,t}(\xi) = \frac{1}{N} \cdot \left(N_t \cdot D_{KL}(y_{t,\min} \| \xi) + \sum_{t'=1}^{T} N_{t'} \cdot H(\boldsymbol{y}_{t'}) \right).$$

This follows since $\frac{\partial D_{KL}(y\|\xi)}{\partial y} > 0$ for any $y > \xi$ and therefore $D_{KL}(y_{it}\|\xi) \geq D_{KL}(y_{t,\min}\|\xi)$ for all $i \in S_t$ and any $\xi \in (0, y_{t,\min}]$.

Now using Claims 1, 2 and 3, it follows that

$$G(\xi) = \min_{1 \leq t \leq T} \frac{1}{N} \cdot \left(N_t \cdot D_{KL}(y_{t,\min} \| \xi) + \sum_{t'=1}^{T} N_{t'} \cdot H(\boldsymbol{y}_{t'}) \right) \quad \text{for any } 0 < \xi \leq y_{\min}.$$

where recall that $y_{\min} = \min_{1 \leq t \leq T} y_{t,\min}$ Given this, it can be verified that:

1. $G(\xi)$ is non-increasing as ξ increases—since we are optimizing over a larger domain.
2. $G(y_{\min}) = \frac{1}{N} \sum_{t'=1}^{T} N_{t'} \cdot H(\boldsymbol{y}_{t'}) \leq F_0 \stackrel{\text{def}}{=} \mathsf{NLL}(\mathbf{g}^{(0)})$ for any initialization $\mathbf{g}^{(0)}$.
3. $G(\xi) \to +\infty$ as $\xi \to 0$.

The above three facts imply that there exists $0 < \xi_{\min} \leq y_{\min}$ such that

$$G(\xi_{\min}) \leq F_0 \quad \text{and} \quad G(\xi_{\min}) > F_0 \text{ for all } 0 < \xi < \xi_{\min} \tag{C.7}$$

This establishes the definition of $\xi_{\min}$ provided in the main text.

Given the above, we claim that for each iterate of the CG algorithm $\mathbf{g}^{(k)}$, $\mathrm{g}_{\min}^{(k)} \geq \xi_{\min}$. Suppose this is not the case, i.e. $\mathrm{g}_{\min}^{(k)} < \xi_{\min}$ for some iterate k. Then, consider the following:

$$\begin{aligned} \mathrm{g}_{\min}^{(k)} < \xi_{\min} &\implies G(\mathrm{g}_{\min}^{(k)}) > F_0 \quad \{\text{from equation (C.7) above}\} \\ &\implies \mathsf{NLL}(\mathbf{g}^{(k)}) > F_0 = \mathsf{NLL}(\mathbf{g}^{(0)}) \end{aligned}$$

where the last implication follows since $\mathbf{g}^{(k)}$ is feasible for problem (C.5) with $\xi = \mathrm{g}_{\min}^{(k)}$ and consequently, $\mathsf{NLL}(\mathbf{g}^{(k)}) \geq G(\mathrm{g}_{\min}^{(k)})$. However, this results in a contradiction since the FCFW variant improves the objective value in each iteration.

Finally, the convergence result follows from choosing $\eta = \xi_{\min}$ in Lemma C.1.

C.1.3 Proof of Proposition 3.2.1

From equation (C.7), it follows that

$$G(\xi_{\min}) \leq F_0 \implies \min_{1 \leq t \leq T} N_t \cdot D_{KL}(y_{t,\min} \| \xi_{\min}) \leq N \cdot F_0 - \sum_{t'=1}^{T} N_{t'} \cdot H(\boldsymbol{y}_{t'}).$$

Now, suppose that $\min_{1\le t\le T} N_t \cdot D_{KL}(y_{t,\min}\|\xi_{\min}) = N_{t^*}\cdot D_{KL}(y_{t^*,\min}\|\xi_{\min})$ for some $1 \le t^* \le T$. Then consider the following:

$$
\begin{aligned}
& N_{t^*}\cdot D_{KL}(y_{t^*,\min}\|\xi_{\min}) \le N\cdot F_0 - \sum_{t=1}^{T} N_t\cdot H(\boldsymbol{y}_t) \\
\iff & D_{KL}(y_{t^*,\min}\|\xi_{\min}) \le \frac{N\cdot F_0 - \sum_{t=1}^{T} N_t\cdot H(\boldsymbol{y}_t)}{N_{t^*}} \\
\iff & y_{t^*,\min}\log\frac{y_{t^*,\min}}{\xi_{\min}} + (1-y_{t^*,\min})\log\frac{1-y_{t^*,\min}}{1-\xi_{\min}} \le \frac{N\cdot F_0 - \sum_{t=1}^{T} N_t\cdot H(\boldsymbol{y}_t)}{N_{t^*}} \\
\iff & y_{t^*,\min}\log\frac{y_{t^*,\min}}{\xi_{\min}} \le \frac{N\cdot F_0 - \sum_{t=1}^{T} N_t\cdot H(\boldsymbol{y}_t)}{N_{t^*}} + (1-y_{t^*,\min})\log\frac{1-\xi_{\min}}{1-y_{t^*,\min}} \\
\implies & y_{t^*,\min}\log\frac{y_{t^*,\min}}{\xi_{\min}} \le \frac{N\cdot F_0 - \sum_{t=1}^{T} N_t\cdot H(\boldsymbol{y}_t)}{N_{t^*}} + (1-y_{t^*,\min})\cdot\left(\frac{1-\xi_{\min}}{1-y_{t^*,\min}} - 1\right) \\
& \quad (\text{since } \log z \le z-1 \ \ \forall\, z>0) \\
\implies & y_{t^*,\min}\log\frac{y_{t^*,\min}}{\xi_{\min}} \le \frac{N\cdot F_0 - \sum_{t=1}^{T} N_t\cdot H(\boldsymbol{y}_t)}{N_{t^*}} + y_{t^*,\min} \\
\iff & \log\frac{y_{t^*,\min}}{\xi_{\min}} \le \frac{N\cdot F_0 - \sum_{t=1}^{T} N_t\cdot H(\boldsymbol{y}_t)}{y_{t^*,\min}\cdot N_{t^*}} + 1 \\
\iff & \frac{y_{t^*,\min}}{\xi_{\min}} \le \exp\left(\frac{N\cdot F_0 - \sum_{t=1}^{T} N_t\cdot H(\boldsymbol{y}_t)}{y_{t^*,\min}\cdot N_{t^*}} + 1\right) \\
\iff & \xi_{\min} \ge y_{t^*,\min}\cdot\exp\left(-1 - \frac{N\cdot F_0 - \sum_{t=1}^{T} N_t\cdot H(\boldsymbol{y}_t)}{y_{t^*,\min}\cdot N_{t^*}}\right) \\
\implies & \xi_{\min} \ge y_{\min}\cdot\exp\left(-1 - \frac{N\cdot F_0 - \sum_{t=1}^{T} N_t\cdot H(\boldsymbol{y}_t)}{y_{t^*,\min}\cdot N_{t^*}}\right) \quad (\text{since } y_{t^*,\min} \ge y_{\min}) \\
\implies & \xi_{\min} \ge y_{\min}\cdot\exp\left(-1 - \frac{N\cdot F_0 - \sum_{t=1}^{T} N_t\cdot H(\boldsymbol{y}_t)}{N_{\min}}\right) \quad (\text{since } y_{t^*,\min}\cdot N_{t^*} \ge N_{\min})
\end{aligned}
$$

and the result follows.

C.2 Proof of Theorem 3.3

In the following, let $\|\cdot\|_2$ and $\|\cdot\|_\infty$ denote the standard ℓ_2 and ℓ_∞ norms on the Euclidean space. We first establish a few key lemmas that will be used in the proof.

Lemma C.2. *Let $\mathbf{A}\in\mathbb{R}^{m\times D}$ and $\mathbf{u}\in\mathbb{R}^m$ be given, for some $m\ge 1$. Suppose there exists sequence $\{\boldsymbol{\omega}_r\}_{r\in\mathbb{N}}$ such that $\lim_{r\to\infty}\|\mathbf{A}\cdot\boldsymbol{\omega}_r - \mathbf{u}\|_\infty = 0$. Then, $\mathbf{u}\in\mathrm{Range}(\mathbf{A})$, i.e. $\mathbf{u}$ lies in the subspace spanned by the columns of the matrix $\mathbf{A}$.*

Proof. First observe that since $0 \leq \|\mathbf{A} \cdot \boldsymbol{\omega}_r - \mathbf{u}\|_2 \leq \sqrt{m} \cdot \|\mathbf{A} \cdot \boldsymbol{\omega}_r - \mathbf{u}\|_\infty$ for all $r \in \mathbb{N}$, it follows from the Squeeze (or Sandwich) theorem that

$$\lim_{r \to \infty} \|\mathbf{A} \cdot \boldsymbol{\omega}_r - \mathbf{u}\|_2 = 0 \tag{C.8}$$

Suppose that $\mathbf{u} \notin \text{Range}(\mathbf{A})$ and let $\mathbf{u} := \mathbf{u}_\parallel + \mathbf{u}_\perp$ where $\mathbf{u}_\parallel$ denote the orthogonal projection of the vector $\mathbf{u}$ onto the subspace Range($\mathbf{A}$), so that $\mathbf{u}_\perp$ is orthogonal to Range($\mathbf{A}$). Since $\mathbf{u} \notin \text{Range}(\mathbf{A})$, we have $\mathbf{u}_\perp \neq \mathbf{0}$.

Then, for any $r \in \mathbb{N}$ it follows that

$$\|\mathbf{A} \cdot \boldsymbol{\omega}_r - \mathbf{u}\|_2^2 = \|\big(\mathbf{A} \cdot \boldsymbol{\omega}_r - \mathbf{u}_\parallel\big) - \mathbf{u}_\perp\|_2^2 = \|\mathbf{A} \cdot \boldsymbol{\omega}_r - \mathbf{u}_\parallel\|_2^2 + \|\mathbf{u}_\perp\|_2^2 \geq \|\mathbf{u}_\perp\|_2^2 > 0.$$

But this contradicts equation (C.8) and therefore $\mathbf{u} \in \text{Range}(\mathbf{A})$. □

Lemma C.3. *Let $\mathbf{A} \in \mathbb{R}^{m \times D}$ and $\mathbf{u} \in \mathbb{R}^m$ be given, for some $m \geq 1$. Suppose that there exists $\boldsymbol{\omega} \in \mathbb{R}^D$ such that $\|\mathbf{A} \cdot \boldsymbol{\omega} - \mathbf{u}\|_\infty < \varepsilon$ for some $\varepsilon > 0$. Let $\Pi_{\mathbf{A}}(\boldsymbol{\omega})$ denote the orthogonal projection of the vector $\boldsymbol{\omega}$ onto the subspace spanned by the* rows *of the matrix $\mathbf{A}$. Then, it follows that $\|\Pi_{\mathbf{A}}(\boldsymbol{\omega})\|_2 \leq \frac{\sqrt{m} \cdot \varepsilon + \|\mathbf{u}\|_2}{\sigma_{\min}(\mathbf{A})}$ where $\sigma_{\min}(\mathbf{A}) > 0$ is the smallest non-zero singular value of the matrix $\mathbf{A}$.*

Proof. Let $D' \leq \min(m, D)$ denote the rank of matrix $\mathbf{A}$. Then, using the singular value decomposition (SVD) of $\mathbf{A}$, we get

$$\mathbf{A} = \mathbf{C}\Sigma\mathbf{R}^\top,$$

where $\mathbf{C} \in \mathbb{R}^{m \times D'}$ is such that its columns represent an orthonormal basis for the column space of $\mathbf{A}$, $\mathbf{R} \in \mathbb{R}^{D \times D'}$ is such that its columns represent an orthonormal basis for the row space of $\mathbf{A}$ and $\Sigma \in \mathbb{R}^{D' \times D'}$ is a diagonal matrix containing the (non-zero) singular values $\sigma_1, \sigma_2, \ldots, \sigma_{D'}$ of the matrix $\mathbf{A}$. Next, represent the vector $\boldsymbol{\omega}$ as $\boldsymbol{\omega} = \boldsymbol{\theta} + \Pi_{\mathbf{A}}(\boldsymbol{\omega})$ where $\boldsymbol{\theta}$ represents the component that is orthogonal to the row space. Since $\Pi_{\mathbf{A}}(\boldsymbol{\omega})$, by definition, lies in the row space, we can write it as $\Pi_{\mathbf{A}}(\boldsymbol{\omega}) = \mathbf{R} \cdot \boldsymbol{\alpha}$ where $\boldsymbol{\alpha} \in \mathbb{R}^{D'}$. Then, it follows that

$$\begin{aligned}
\|\mathbf{A} \cdot \boldsymbol{\omega} - \mathbf{u}\|_\infty < \varepsilon &\implies \|\mathbf{A} \cdot \boldsymbol{\omega} - \mathbf{u}\|_2 < \sqrt{m} \cdot \varepsilon \\
&\implies \|\mathbf{A} \cdot \boldsymbol{\omega}\|_2 < \sqrt{m} \cdot \varepsilon + \|\mathbf{u}\|_2 \quad \{\text{by (reverse) triangle inequality}\} \\
&\iff \|\mathbf{A} \cdot \Pi_{\mathbf{A}}(\boldsymbol{\omega})\|_2 < \sqrt{m} \cdot \varepsilon + \|\mathbf{u}\|_2 \\
&\qquad (\text{since } \boldsymbol{\omega} = \boldsymbol{\theta} + \Pi_{\mathbf{A}}(\boldsymbol{\omega}) \text{ and } \mathbf{A} \cdot \boldsymbol{\theta} = \mathbf{0})
\end{aligned}$$

Next, consider the following:

$$\begin{aligned}
\|\mathbf{A} \cdot \Pi_{\mathbf{A}}(\boldsymbol{\omega})\|_2 &= \|\mathbf{A} \cdot (\mathbf{R} \cdot \boldsymbol{\alpha})\|_2 \\
&= \|\mathbf{C}\Sigma\mathbf{R}^\top \cdot (\mathbf{R} \cdot \boldsymbol{\alpha})\|_2 \quad \text{(using SVD of } \mathbf{A}\text{)} \\
&= \|(\mathbf{C}\Sigma) \cdot \boldsymbol{\alpha}\|_2 \quad \text{(since columns of } \mathbf{R} \text{ are orthonormal)} \\
&= \|\Sigma \cdot \boldsymbol{\alpha}\|_2 \quad \text{(since } \|\mathbf{C} \cdot \mathbf{x}\| = \|\mathbf{x}\| \text{ for any } \mathbf{x} \text{ as } \mathbf{C} \text{ is unitary)} \\
&= \sqrt{\sum_{d=1}^{D'} \sigma_d^2 \alpha_d^2} \\
&\geq \sigma_{\min}(\mathbf{A}) \cdot \|\boldsymbol{\alpha}\| \\
&= \sigma_{\min}(\mathbf{A}) \cdot \|\mathbf{R} \cdot \boldsymbol{\alpha}\| \quad \text{(since } \mathbf{R} \text{ is unitary)} \\
&= \sigma_{\min}(\mathbf{A}) \cdot \|\Pi_{\mathbf{A}}(\boldsymbol{\omega})\|
\end{aligned}$$

The claim then follows. □

Next, w.l.o.g suppose that the product features remain fixed across the offer-sets, i.e. $\mathbf{z}_{jt} = \mathbf{z}_{jt'}$ for all $j \in [n]$ and all $t \neq t'$—if features are varying across offer-sets, we can just expand the product universe $[n]$. We represent the features as $\{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n\}$ in the remainder.

Let $\boldsymbol{f} \in \overline{\mathcal{P}} \setminus \mathcal{P}$ be any boundary type. Since $\mathbb{R}^M$ is a metric space, $\overline{\mathcal{P}}$ is precisely the set of the limits of all convergent sequences in $\mathcal{P}$. Therefore, there exists a sequence $\{\boldsymbol{\omega}_r\}_{r \in \mathbb{N}} \subset \mathbb{R}^D$ such that $\lim_{r \to \infty} \boldsymbol{f}(\boldsymbol{\omega}_r) = \boldsymbol{f}$. In addition, it follows that there exists a permutation $\pi : [n] \to [n]$ such that there is a subsequence $\{\boldsymbol{\omega}_{r_\ell}\}_{\ell \in \mathbb{N}}$ satisfying $\boldsymbol{\omega}_{r_\ell}^\top \mathbf{z}_{\pi(1)} \geq \boldsymbol{\omega}_{r_\ell}^\top \mathbf{z}_{\pi(2)} \geq \ldots \geq \boldsymbol{\omega}_{r_\ell}^\top \mathbf{z}_{\pi(n)}$ for all $\ell \in \mathbb{N}$ (this is because the set of permutations of n elements is finite). Since every subsequence must converge to the same limit, we must have

$$\lim_{\ell \to \infty} \boldsymbol{f}(\boldsymbol{\omega}_{r_\ell}) = \boldsymbol{f}.$$

For brevity of notation, we refer to the sequence $\{\boldsymbol{\omega}_{r_\ell}\}_{\ell \in \mathbb{N}}$ as the sequence $\{\boldsymbol{\omega}_r\}_{r \in \mathbb{N}}$ in the remainder, and w.l.o.g assume that the products are indexed such that $\boldsymbol{\omega}_r^\top \mathbf{z}_1 \geq \boldsymbol{\omega}_r^\top \mathbf{z}_2 \geq \ldots \geq \boldsymbol{\omega}_r^\top \mathbf{z}_n$.

We then establish the following key lemma:

Lemma C.4. *Consider any offer-set S_t. Let $i_t = \arg\min_{j \in S_t} j$, i.e. i_t is the product with the minimum index in S_t. For any $j \in S_t$, it follows that*

1. *If $f_{jt} = 0$, then $\lim_{r \to \infty} \boldsymbol{\omega}_r^\top (\mathbf{z}_{i_t} - \mathbf{z}_j) = +\infty$.*
2. *If $f_{jt} > 0$, then $\lim_{r \to \infty} \boldsymbol{\omega}_r^\top (\mathbf{z}_{i_t} - \mathbf{z}_j) = u_{i_t j} \geq 0$ for some finite $u_{i_t j}$.*

Proof. Note that since $\boldsymbol{\omega}_r^\top \mathbf{z}_{i_t} \geq \boldsymbol{\omega}_r^\top \mathbf{z}_j$ for all $r \in \mathbb{N}$, it follows that $f_{i_t t} \geq f_{jt}$ for all $j \in S_t$. Further, note that $f_{i_t t} > 0$ otherwise $f_{jt} = 0$ for all $j \in S_t$ which is a contradiction since

choice probabilities within each offer-set must sum to 1. Now for any $j \in S_t$, consider the following:

$$\exp\left(\boldsymbol{\omega}_r^\top(\mathbf{z}_j - \mathbf{z}_{i_t})\right) = \frac{f_{jt}(\boldsymbol{\omega}_r)}{f_{i_t t}(\boldsymbol{\omega}_r)}$$
$$\implies \lim_{r\to\infty} \exp\left(\boldsymbol{\omega}_r^\top(\mathbf{z}_j - \mathbf{z}_{i_t})\right) = \lim_{r\to\infty} \frac{f_{jt}(\boldsymbol{\omega}_r)}{f_{i_t t}(\boldsymbol{\omega}_r)} = \frac{\lim_{r\to\infty} f_{jt}(\boldsymbol{\omega}_r)}{\lim_{r\to\infty} f_{i_t t}(\boldsymbol{\omega}_r)} = \frac{f_{jt}}{f_{i_t t}}.$$

From the above, it follows that if $f_{jt} = 0$, then $\lim_{r\to\infty} \exp\left(\boldsymbol{\omega}_r^\top(\mathbf{z}_j - \mathbf{z}_{i_t})\right) = 0$ or equivalently, $\lim_{r\to\infty} \boldsymbol{\omega}_r^\top(\mathbf{z}_j - \mathbf{z}_{i_t}) = -\infty$. When $f_{jt} > 0$, then since $\log(\cdot)$ is continuous, it follows that $\lim_{r\to\infty} \boldsymbol{\omega}_r^\top(\mathbf{z}_j - \mathbf{z}_{i_t}) = \log \frac{f_{jt}}{f_{i_t t}} \leq 0$ because $f_{jt} \leq f_{i_t t}$ for all $j \in S_t$. The claim then follows.

Finally, note that the same pair of products (i, j) could appear in two different offer-sets, but the uniqueness of limits ensures that the sequence $\boldsymbol{\omega}_r^\top(\mathbf{z}_i - \mathbf{z}_j)$ will converge to the same quantity in both cases. □

We are now ready to prove the result. We first need some additional notation. Let $\text{Pairs}_t = \{(i_t, j) \mid j \in S_t \setminus \{i_t\}$ and $\lim_{r\to\infty} \boldsymbol{\omega}_r^\top(\mathbf{z}_{i_t} - \mathbf{z}_j) < \infty\}$, note that Pairs_t could be empty for any $1 \leq t \leq T$. Denote $\text{Pairs} = \cup_{t=1}^T \text{Pairs}_t$. Similarly, define the set $\overline{\text{Pairs}}_t = \{(i_t, j) \mid j \in S_t \setminus \{i_t\}$ and $\lim_{r\to\infty} \boldsymbol{\omega}_r^\top(\mathbf{z}_{i_t} - \mathbf{z}_j) = +\infty\}$ and denote $\overline{\text{Pairs}} = \cup_{t=1}^T \overline{\text{Pairs}}_t$. Note that $\overline{\text{Pairs}}_t$ could also be empty for some $t \in [T]$, but we make the following claim:

Claim 1:
$$\exists t' \in [T] \text{ such that } \overline{\text{Pairs}}_{t'} \neq \emptyset.$$

Suppose this is not true, so that $\overline{\text{Pairs}} = \cup_{t=1}^T \overline{\text{Pairs}}_t = \emptyset$. This means that $\text{Pairs} \neq \emptyset$ since, by definition, each product pair must belong to either Pairs or $\overline{\text{Pairs}}$. Then, construct the matrix $\mathbf{A}^{\text{Pairs}} \in \mathbb{R}^{|\text{Pairs}| \times D}$ where row $\mathbf{a}_{ij}^{\text{Pairs}}$ corresponding to pair $(i, j) \in \text{Pairs}$ is given by $\mathbf{a}_{ij}^{\text{Pairs}} = \mathbf{z}_i - \mathbf{z}_j$. Similarly, let $\mathbf{u} \in \mathbb{R}^{|\text{Pairs}|}$ denote the vector of utilities u_{ij} for each pair $(i, j) \in \text{Pairs}$ (refer to Lemma C.4). Then it follows that $\lim_{r\to\infty} \|\mathbf{A}^{\text{Pairs}} \cdot \boldsymbol{\omega}_r - \mathbf{u}\|_\infty = 0$.

Now, applying Lemma C.2 tells us that $\mathbf{u} \in \text{Range}(\mathbf{A}^{\text{Pairs}})$, i.e. there exists $\boldsymbol{\omega}_0 \in \mathbb{R}^D$ such that $\mathbf{A}^{\text{Pairs}} \cdot \boldsymbol{\omega}_0 = \mathbf{u}$. Then, from Lemma C.4 it follows that for any $j \in S_t$ and any $1 \leq t \leq T$:

$$\begin{aligned} f_{jt} &= \exp(-u_{i_t j}) \cdot f_{i_t t} \quad (\text{since } f_{jt} > 0) \\ &= \exp\left(\boldsymbol{\omega}_0^\top(\mathbf{z}_j - \mathbf{z}_{i_t})\right) \cdot f_{i_t t} \quad \left(\text{since } u_{i_t j} = (\mathbf{z}_{i_t} - \mathbf{z}_j)^\top \boldsymbol{\omega}_0 \text{ as shown above}\right) \\ \implies f_{jt} &= \frac{\exp\left(\boldsymbol{\omega}_0^\top(\mathbf{z}_j - \mathbf{z}_{i_t})\right)}{1 + \sum_{\ell \in S_t \setminus \{i_t\}} \exp\left(\boldsymbol{\omega}_0^\top(\mathbf{z}_\ell - \mathbf{z}_{i_t})\right)} \qquad \left(\text{since } \sum_{\ell \in S_t} f_{\ell t} = 1\right) \\ \iff f_{jt} &= \frac{\exp\left(\boldsymbol{\omega}_0^\top \mathbf{z}_j\right)}{\sum_{\ell \in S_t} \exp\left(\boldsymbol{\omega}_0^\top \mathbf{z}_\ell\right)} = f_{jt}(\boldsymbol{\omega}_0) \end{aligned}$$

That is, $\boldsymbol{f} = \boldsymbol{f}(\boldsymbol{\omega}_0)$. But since $\boldsymbol{f}(\boldsymbol{\omega}_0) \in \mathcal{P}$ by definition, this means that $\boldsymbol{f} \in \mathcal{P}$ which contradicts the assumption that $\boldsymbol{f}$ is a boundary type and belongs to $\overline{\mathcal{P}} \setminus \mathcal{P}$. The claim then follows. In addition, if $\overline{\text{Pairs}}_{t'} \neq \emptyset$, then for any pair $(i_{t'}, j') \in \overline{\text{Pairs}}_{t'}$, it follows from Lemma C.4 that $f_{j' t'} = 0$ establishing the second part of the theorem.

Following Claim 1, there are two cases which we deal with separately:

Case 1: Pairs $= \emptyset$. In this case, it follows from Lemma C.4 that $\boldsymbol{f}$ is a boundary type that chooses only a single product, viz. i_t from each offer-set S_t. Based on the definition of $\overline{\text{Pairs}}$, we know that for any $U > 0$, there exists $\tilde{\boldsymbol{\omega}}$ such that $\tilde{\boldsymbol{\omega}}^\top(\mathbf{z}_{\bar{i}} - \mathbf{z}_{\bar{j}}) > U$ for all $(\bar{i}, \bar{j}) \in \overline{\text{Pairs}}$. Then, it follows that the choice probabilities under the boundary type $\boldsymbol{f}$ are equal to the limiting choice probabilities using $\boldsymbol{\omega}_0 = \mathbf{0}$ (i.e. the all zeros vector) and $\boldsymbol{\theta} = \frac{\tilde{\boldsymbol{\omega}}}{\|\tilde{\boldsymbol{\omega}}\|}$.

Case 2: Pairs $\neq \emptyset$. In this case, first we construct the matrix $\mathbf{A}^{\text{Pairs}} \in \mathbb{R}^{|\text{Pairs}| \times D}$ as outlined above in the proof of Claim 1. Given this, we choose the parameters $(\boldsymbol{\omega}_0, \boldsymbol{\theta})$ as follows:

Choosing $\boldsymbol{\omega}_0$. From Lemma C.4 above, it follows that $\lim_{r\to\infty} \|\mathbf{A}^{\text{Pairs}} \cdot \boldsymbol{\omega}_r - \mathbf{u}\|_\infty = 0$. Then, Lemma C.2 tells us that $\mathbf{u} \in \text{Range}(\mathbf{A}^{\text{Pairs}})$, i.e. there exists $\boldsymbol{\omega}_0 \in \mathbb{R}^D$ such that $\mathbf{A}^{\text{Pairs}} \cdot \boldsymbol{\omega}_0 = \mathbf{u}$.

Choosing $\boldsymbol{\theta}$. Next, using the definition of Pairs and $\overline{\text{Pairs}}$, it follows that given *any* $\varepsilon > 0$ and $U > 0$, there exists $\tilde{\boldsymbol{\omega}}$ such that:

$$\tilde{\boldsymbol{\omega}}^\top(\mathbf{z}_i - \mathbf{z}_j) < u_{ij} + \varepsilon \;\; \forall\, (i,j) \in \text{Pairs} \;\text{ and }\; \tilde{\boldsymbol{\omega}}^\top(\mathbf{z}_{\bar{i}} - \mathbf{z}_{\bar{j}}) > U \;\; \forall\, (\bar{i}, \bar{j}) \in \overline{\text{Pairs}}.$$

Fix some $\varepsilon > 0$. Then, the choice of $\tilde{\boldsymbol{\omega}}$ implies that $\|\mathbf{A}^{\text{Pairs}} \cdot \tilde{\boldsymbol{\omega}} - \mathbf{u}\|_\infty < \varepsilon$ so that we can apply Lemma C.3 to establish that

$$\|\Pi_{\mathbf{A}^{\text{Pairs}}}(\tilde{\boldsymbol{\omega}})\|_2 < \frac{\sqrt{|\text{Pairs}|} \cdot \varepsilon + \|\mathbf{u}\|_2}{\sigma_{\min}(\mathbf{A}^{\text{Pairs}})}. \tag{C.9}$$

Choose U such that

$$U > \frac{\sqrt{|\text{Pairs}|} \cdot \varepsilon + \|\mathbf{u}\|_2}{\sigma_{\min}(\mathbf{A}^{\text{Pairs}})} \cdot B,$$

where $B \overset{\text{def}}{=} \max_{(\bar{i},\bar{j}) \in \overline{\text{Pairs}}} \|\mathbf{z}_{\bar{i}} - \mathbf{z}_{\bar{j}}\|_2$. Next, choose $\boldsymbol{\theta} = \tilde{\boldsymbol{\omega}} - \Pi_{\mathbf{A}^{\text{Pairs}}}(\tilde{\boldsymbol{\omega}})$ where $\Pi_{\mathbf{A}^{\text{Pairs}}}(\tilde{\boldsymbol{\omega}})$ is the projection of $\tilde{\boldsymbol{\omega}}$ onto the subspace spanned by the rows of $\mathbf{A}^{\text{Pairs}}$. We show that $\boldsymbol{\theta}$ satisfies:

$$(1)\; \boldsymbol{\theta}^\top(\mathbf{z}_i - \mathbf{z}_j) = 0 \;\; \forall\, (i,j) \in \text{Pairs} \;\text{ and }\; (2)\; \boldsymbol{\theta}^\top(\mathbf{z}_{\bar{i}} - \mathbf{z}_{\bar{j}}) > 0 \;\; \forall\, (\bar{i}, \bar{j}) \in \overline{\text{Pairs}}.$$

Part (1) follows since $\boldsymbol{\theta}$ is orthogonal to the subspace spanned by the rows of $\mathbf{A}^{\text{Pairs}}$. For part (2), consider the following, for any $(\bar{i}, \bar{j}) \in \overline{\text{Pairs}}$:

$$\begin{aligned}
&\tilde{\boldsymbol{\omega}}^\top(\mathbf{z}_{\bar{i}} - \mathbf{z}_{\bar{j}}) > U \\
\iff &\left(\boldsymbol{\theta} + \Pi_{\mathbf{A}^{\text{Pairs}}}(\tilde{\boldsymbol{\omega}})\right)^\top (\mathbf{z}_{\bar{i}} - \mathbf{z}_{\bar{j}}) > U \\
\iff &\boldsymbol{\theta}^\top(\mathbf{z}_{\bar{i}} - \mathbf{z}_{\bar{j}}) + \Pi_{\mathbf{A}^{\text{Pairs}}}(\tilde{\boldsymbol{\omega}})^\top(\mathbf{z}_{\bar{i}} - \mathbf{z}_{\bar{j}}) > U \\
\implies &\boldsymbol{\theta}^\top(\mathbf{z}_{\bar{i}} - \mathbf{z}_{\bar{j}}) + \|\Pi_{\mathbf{A}^{\text{Pairs}}}(\tilde{\boldsymbol{\omega}})\|_2 \cdot \|\mathbf{z}_{\bar{i}} - \mathbf{z}_{\bar{j}}\|_2 > U \quad \text{(using Cauchy-Schwarz inequality)} \\
\iff &\boldsymbol{\theta}^\top(\mathbf{z}_{\bar{i}} - \mathbf{z}_{\bar{j}}) > U - \|\Pi_{\mathbf{A}^{\text{Pairs}}}(\tilde{\boldsymbol{\omega}})\|_2 \cdot \|\mathbf{z}_{\bar{i}} - \mathbf{z}_{\bar{j}}\|_2 \\
\implies &\boldsymbol{\theta}^\top(\mathbf{z}_{\bar{i}} - \mathbf{z}_{\bar{j}}) > U - \|\Pi_{\mathbf{A}^{\text{Pairs}}}(\tilde{\boldsymbol{\omega}})\|_2 \cdot B \quad \left(\text{since } \|\mathbf{z}_{\bar{i}} - \mathbf{z}_{\bar{j}}\|_2 \le B \text{ by definition of } B\right) \\
\implies &\boldsymbol{\theta}^\top(\mathbf{z}_{\bar{i}} - \mathbf{z}_{\bar{j}}) > U - \frac{\sqrt{|\text{Pairs}|} \cdot \varepsilon + \|\mathbf{u}\|_2}{\sigma_{\min}(\mathbf{A}^{\text{Pairs}})} \cdot B \quad \{\text{using equation (C.9)}\} \\
\implies &\boldsymbol{\theta}^\top(\mathbf{z}_{\bar{i}} - \mathbf{z}_{\bar{j}}) > 0 \quad \text{(by choice of } U)
\end{aligned}$$

Then, it is easy to see that the choice probabilities under the boundary type $\boldsymbol{f}$ are equal to the limiting probabilities for the choice $(\boldsymbol{\omega}_0, \boldsymbol{\theta})$ computed above.

C.3 Proof of Theorem 3.4

Define the function $H : \overline{\mathcal{P}} \to \mathbb{R}$ such that $H(\boldsymbol{f}) = \sum_{i=1}^{n} c_i f_i$ for each $\boldsymbol{f} \in \overline{\mathcal{P}}$. Consequently, the support finding step (3.5) can be equivalently written as:

$$\max_{\boldsymbol{f} \in \overline{\mathcal{P}}} \quad H(\boldsymbol{f}) \tag{C.10}$$

Let $C^* = \max_{\boldsymbol{f} \in \overline{\mathcal{P}}} H(\boldsymbol{f})$ denote the optimal objective of the above subproblem, note that this is well-defined since $H(\cdot)$ is continuous and $\overline{\mathcal{P}}$ is compact.

Without loss of generality, index the products such that $c_1 \geq c_2 \geq \cdots \geq c_n$. Note that $C^* \leq c_1$ because the objective value in subproblem (C.10) is a convex combination of $\{c_1, c_2, \cdots, c_n\}$. Since $\mathbf{z}_1$ is an extreme point, it follows from Lemma C.6 (proved below) that $\mathbf{e}_1 \in \overline{\mathcal{P}}$, where $\mathbf{e}_1$ is defined as:

$$e_{1i} = \begin{cases} 1 & \text{if } i = 1 \\ 0 & \text{otherwise.} \end{cases}$$

Then it follows that

$$H(\mathbf{e}_1) = \sum_{i=1}^{n} c_i e_{1i} = c_1.$$

In addition, Lemma C.5 below shows the existence of $\boldsymbol{\theta}_1 \in \mathbb{R}^D$ such that $\boldsymbol{\theta}_1^\top \mathbf{z}_1 > \boldsymbol{\theta}_1^\top \mathbf{z}_i$ for all $1 < i \leq n$. Then, it is easy to see that $\mathbf{e}_1 = \boldsymbol{f}(\mathbf{0}, \boldsymbol{\theta}_1)$ from which the result follows.

To complete the proof, we now establish the two lemmas referenced above. The first provides a characterization of extreme points of the polytope $\mathcal{Z}_n$:

Lemma C.5 (Characterization of extreme points)**.** *$\mathbf{z}_j$ is an extreme point of the polytope $\mathcal{Z}_n$ if and only if there exists $\boldsymbol{\theta} \in \mathbb{R}^D$ such that $\boldsymbol{\theta}^\top \mathbf{z}_j > \boldsymbol{\theta}^\top \mathbf{z}_i$ for all $i \neq j$.*

Proof. **"if" direction**. If possible, suppose that $\mathbf{z}_j$ is not an extreme point, i.e. $\mathbf{z}_j \in \text{conv}(\{\mathbf{z}_i \ : \ i \neq j\})$ so that there exists coefficients $\lambda_{ij} \geq 0$ such that

$$\mathbf{z}_j = \sum_{i \neq j} \lambda_{ij} \mathbf{z}_i \ ; \quad \sum_{i \neq j} \lambda_{ij} = 1 \ ; \quad \lambda_{ij} \geq 0 \quad \forall \ i \neq j.$$

But this results in the following contradiction:

$$\begin{aligned}
\mathbf{z}_j = \sum_{i \neq j} \lambda_{ij} \mathbf{z}_i \implies \boldsymbol{\theta}^\top \mathbf{z}_j &= \sum_{i \neq j} \lambda_{ij} \boldsymbol{\theta}^\top \mathbf{z}_i \\
\implies \boldsymbol{\theta}^\top \mathbf{z}_j &< \sum_{i \neq j} \lambda_{ij} \boldsymbol{\theta}^\top \mathbf{z}_j \quad (\text{since } \lambda_{ij} > 0 \text{ for some } i) \\
\implies \boldsymbol{\theta}^\top \mathbf{z}_j &< \boldsymbol{\theta}^\top \mathbf{z}_j \cdot \left(\sum_{i \neq j} \lambda_{ij} \right) \\
\implies \boldsymbol{\theta}^\top \mathbf{z}_j &< \boldsymbol{\theta}^\top \mathbf{z}_j
\end{aligned}$$

"only if" direction. Denote $\mathcal{M} = \text{conv}(\{\mathbf{z}_i \ : \ i \neq j\})$ and observe that $\mathcal{M}$ is a closed, convex and proper subset of $\mathbb{R}^D$. Now since $\mathbf{z}_j \notin \mathcal{M}$, it follows from the (strong) separation theorem in convex analysis that there exists $\boldsymbol{\theta} \in \mathbb{R}^D$ such that

$$\boldsymbol{\theta}^\top \mathbf{z}_j > \boldsymbol{\theta}^\top \mathbf{z}_i \ \ \forall \ \ i \neq j.$$

□

Next, we show that for each product feature vector that is an extreme point, there exists a boundary type that chooses the product with probability 1 from offer-set $[n]$:

Lemma C.6. *Suppose $\mathbf{z}_j$ is an extreme point of the polytope $\mathcal{Z}_n$. Define the vector $\mathbf{e}_j = (e_{j1}, e_{j2}, \ldots, e_{jn})$ as follows:*

$$e_{ji} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases}$$

Then $\mathbf{e}_j \in \overline{\mathcal{P}}$, in other words, there exists a boundary type that chooses product j with probability 1 from offer-set $[n]$.

Proof. Since $\mathbf{z}_j$ is an extreme point, it follows from Lemma C.5 that

$$\exists \boldsymbol{\theta} \in \mathbb{R}^D \ \text{ s.t. } \ \boldsymbol{\theta}^\top \mathbf{z}_j > \boldsymbol{\theta}^\top \mathbf{z}_i \quad \forall i \neq j.$$

Consider the sequence $\{r \cdot \boldsymbol{\theta}\}_{r \in \mathbb{N}} \subseteq \mathbb{R}^D$. For any $i \neq j$, note that:

$$\begin{aligned}
\lim_{r \to \infty} f_i(r \cdot \boldsymbol{\theta}) = \lim_{r \to \infty} \frac{\exp\left(r \cdot \boldsymbol{\theta}^\top \mathbf{z}_i\right)}{\sum_{\ell=1}^n \exp\left(r \cdot \boldsymbol{\theta}^\top \mathbf{z}_\ell\right)} &= \lim_{r \to \infty} \frac{\exp\left(-r \cdot (\boldsymbol{\theta}^\top \mathbf{z}_j - \boldsymbol{\theta}^\top \mathbf{z}_i)\right)}{1 + \sum_{\ell \neq j} \exp\left(-r \cdot (\boldsymbol{\theta}^\top \mathbf{z}_j - \boldsymbol{\theta}^\top \mathbf{z}_\ell)\right)} \\
&= \frac{0}{1+0} = 0.
\end{aligned}$$

An analogous argument shows that

$$\lim_{r \to \infty} f_j(r \cdot \boldsymbol{\theta}) = \lim_{r \to \infty} \frac{\exp\left(r \cdot \boldsymbol{\theta}^\top \mathbf{z}_j\right)}{\sum_{\ell=1}^n \exp\left(r \cdot \boldsymbol{\theta}^\top \mathbf{z}_\ell\right)} = 1.$$

From the above statements it follows that

$$\lim_{r\to\infty} \boldsymbol{f}(r\cdot\boldsymbol{\theta}) = \mathbf{e}_j,$$

and since the closure contains all limit points of convergent sequences, it follows that $\mathbf{e}_j \in \overline{\mathcal{P}}$. □

C.4 Proof of Theorem 3.5

Since each $\mathbf{z}_j$ is an extreme point, it follows from Lemma C.6 that $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n \in \overline{\mathcal{P}}$. Then it follows that $\boldsymbol{y} \in \text{conv}(\overline{\mathcal{P}})$ since $\boldsymbol{y} = \sum_{i=1}^n y_i \cdot \mathbf{e}_i$ and $\sum_{i=1}^n y_i = 1$. Further, it is easy to see that $\mathsf{SQ}(\boldsymbol{y}) = 0$ and $\mathsf{SQ}(\mathbf{g}) > 0$ for all $\mathbf{g} \neq \boldsymbol{y}$. Similarly, we have that $\mathsf{NLL}(\boldsymbol{y}) = -\sum_{i=1}^n y_i \log y_i$ and it can be shown that $\mathsf{NLL}(\mathbf{g}) > \mathsf{NLL}(\boldsymbol{y})$ for all $\mathbf{g} \neq \boldsymbol{y}$ (using the fact that relative entropy or KL-divergence is non-negative). Therefore $\mathbf{g}^* = \boldsymbol{y}$ for both the squared and negative log-likelihood loss functions.

Now, if at some iteration $1 \leq k \leq n$, we have $\left\langle \nabla\mathsf{loss}(\mathbf{g}^{(k-1)}), \boldsymbol{f}^{(k)} - \mathbf{g}^{(k-1)} \right\rangle \geq 0$, then by convexity it follows that $\mathsf{loss}(\mathbf{g}) \geq \mathsf{loss}(\mathbf{g}^{(k-1)})$ for all $\mathbf{g} \in \text{conv}(\overline{\mathcal{P}})$ which means that $\mathbf{g}^{(k-1)} = \mathbf{g}^*$ and the Algorithm terminates. So, suppose that $\left\langle \nabla\mathsf{loss}(\mathbf{g}^{(k-1)}), \boldsymbol{f}^{(k)} - \mathbf{g}^{(k-1)} \right\rangle < 0$ for each $1 \leq k \leq n$, which means that $\boldsymbol{f}^{(k)} - \mathbf{g}^{(k-1)}$ is a descent direction and an improving solution can be found. The proportions update step in Algorithm 8 ensures that we have $\boldsymbol{f}^{(k_1)} \neq \boldsymbol{f}^{(k_2)}$ in any two iterations $k_1 \neq k_2$ (since it optimizes over all previously found types). In addition, Theorem 3.4 shows that we recover only boundary types in each iteration, from which it follows that at the end of n iterations, we have found the types $\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_n$. Now, clearly $\sum_{i=1}^n y_i \cdot \mathbf{e}_i = \boldsymbol{y} = \mathbf{g}^*$ and since the proportions update step optimizes over the convex hull of all previously found types, the claim follows.

C.5 Proof of Theorem 3.6

For ease of exposition, we prove the result for the case when $D_2 = 1$, i.e. there is only a single binary feature but the proof can be easily extended, albeit with additional notation, to the general case. Define the function $G(\cdot) : \mathbb{R}^{D_1+1} \to \mathbb{R}$ for $\boldsymbol{\omega} \in \mathbb{R}^{D_1}$ and $\delta \in \mathbb{R}$ as:

$$G(\boldsymbol{\omega}, \delta) = \frac{\sum_{i=1}^n c_i \exp\left(\boldsymbol{\omega}^\top \mathbf{z}_i + \delta \cdot b_i\right)}{\sum_{j=1}^n \exp\left(\boldsymbol{\omega}^\top \mathbf{z}_j + \delta \cdot b_j\right)}.$$

Further, let $G^* = \sup_{\boldsymbol{\omega}\in\mathbb{R}^{D_1}, \delta\in\mathbb{R}} G(\boldsymbol{\omega}, \delta)$; since $G(\cdot)$ is bounded above, G^* is finite. Note that in this case $\mathcal{P} = \{\boldsymbol{f}(\boldsymbol{\omega}, \delta) \ : \ \boldsymbol{\omega} \in \mathbb{R}^{D_1}, \delta \in \mathbb{R}\}$.[1]

Let S_0 be the set of products that have the binary feature absent, i.e. $S_0 = \{i \in [n] \ : \ b_i = 0\}$ and let $S_1 = [n] \setminus S_0$. For $e \in \{0, 1\}$, define the sets $\mathcal{P}_e = \{\boldsymbol{f}^{(e)}(\boldsymbol{\omega}) \ : \ \boldsymbol{\omega} \in \mathbb{R}^{D_1}\} \subseteq [0, 1]^{|S_e|}$

[1] To simplify notation, we denote $\boldsymbol{f}((\boldsymbol{\omega}, \delta))$ as $\boldsymbol{f}(\boldsymbol{\omega}, \delta)$.

where $\boldsymbol{f}^{(e)}(\boldsymbol{\omega}) = \left(f_i^{(e)}(\boldsymbol{\omega})\right)_{i\in S_e}$ and

$$f_i^{(e)}(\boldsymbol{\omega}) = \frac{\exp\left(\boldsymbol{\omega}^\top \mathbf{z}_i\right)}{\sum_{j\in S_e}\exp\left(\boldsymbol{\omega}^\top \mathbf{z}_j\right)} \tag{C.11}$$

In other words, $\mathcal{P}_0$ (resp. $\mathcal{P}_1$) corresponds to choice probabilities under boundary types that do not consider any product in S_1 (resp. S_0). Let $\boldsymbol{f}^{(0)}, \boldsymbol{f}^{(1)}$ denote arbitrary elements in $\overline{\mathcal{P}}_0$ and $\overline{\mathcal{P}}_1$ where $\overline{\mathcal{P}}_0$ and $\overline{\mathcal{P}}_1$ denote the closures of the sets $\mathcal{P}_0$ and $\mathcal{P}_1$ respectively. Then, consider the following optimization problems for $e \in \{0,1\}$:

$$(\mathrm{P}_e): \quad \underset{\boldsymbol{f}^{(e)}\in\overline{\mathcal{P}}_e}{\arg\max}\ H_e(\boldsymbol{f}^{(e)}),$$

where $H_e : \overline{\mathcal{P}}_e \to \mathbb{R}$ is such that $H_e(\boldsymbol{f}^{(e)}) = \sum_{i\in S_e} c_i f_i^{(e)}$.

Next, for any $\boldsymbol{\omega} \in \mathbb{R}^{D_1}$, define the following:

$$a_0(\boldsymbol{\omega}) = \sum_{j\in S_0} c_j \exp\left(\boldsymbol{\omega}^\top \mathbf{z}_j\right);\ \ b_0(\boldsymbol{\omega}) = \sum_{j\in S_0} \exp\left(\boldsymbol{\omega}^\top \mathbf{z}_j\right);\quad G_0(\boldsymbol{\omega}) = \frac{a_0(\boldsymbol{\omega})}{b_0(\boldsymbol{\omega})}.$$

$$a_1(\boldsymbol{\omega}) = \sum_{j\in S_1} c_j \exp\left(\boldsymbol{\omega}^\top \mathbf{z}_j\right);\ \ b_1(\boldsymbol{\omega}) = \sum_{j\in S_1} \exp\left(\boldsymbol{\omega}^\top \mathbf{z}_j\right);\quad G_1(\boldsymbol{\omega}) = \frac{a_1(\boldsymbol{\omega})}{b_1(\boldsymbol{\omega})}.$$

Further, let $C^*_{(0)} = \sup_{\boldsymbol{\omega}\in\mathbb{R}^{D_1}} G_0(\boldsymbol{\omega})$ and similarly, $C^*_{(1)} = \sup_{\boldsymbol{\omega}\in\mathbb{R}^{D_1}} G_1(\boldsymbol{\omega})$. Recall that $C^* = \max_{\boldsymbol{f}\in\overline{\mathcal{P}}} H(\boldsymbol{f})$.

Claim 1:

$$(1)\ C^* = G^*; \quad (2)\ C^*_{(0)} = \max_{\boldsymbol{f}^{(0)}\in\overline{\mathcal{P}}_0} H_0(\boldsymbol{f}^{(0)}) \text{ and } C^*_{(1)} = \max_{\boldsymbol{f}^{(1)}\in\overline{\mathcal{P}}_1} H_1(\boldsymbol{f}^{(1)}).$$

Consider part (1). First observe that, $G(\boldsymbol{\omega},\delta) = H\left(\boldsymbol{f}(\boldsymbol{\omega},\delta)\right) \le C^*$ for all $\boldsymbol{\omega} \in \mathbb{R}^{D_1}, \delta \in \mathbb{R}$. As supremum is the least upper bound, it follows that $G^* \le C^*$. Next, for *any* $\boldsymbol{f} \in \overline{\mathcal{P}}$, there exists a sequence $\{(\boldsymbol{\omega}_r,\delta_r)\}_{r\in\mathbb{N}} \subset \mathbb{R}^{D_1+1}$ such that $\lim_{r\to\infty} \boldsymbol{f}(\boldsymbol{\omega}_r,\delta_r) = \boldsymbol{f}$. Then, since $H(\cdot)$ is continuous, it follows that

$$\boldsymbol{f} = \lim_{r\to\infty} \boldsymbol{f}(\boldsymbol{\omega}_r,\delta_r) \implies H(\boldsymbol{f}) = \lim_{r\to\infty} H\left(\boldsymbol{f}(\boldsymbol{\omega}_r,\delta_r)\right) = \lim_{r\to\infty} G(\boldsymbol{\omega}_r,\delta_r) \le G^*,$$

where the last inequality follows since $G(\boldsymbol{\omega}_r,\delta_r) \le G^*$ for all $r \in \mathbb{N}$. Since $\boldsymbol{f} \in \overline{\mathcal{P}}$ was arbitrary, this means that $H(\boldsymbol{f}) \le G^*$ for all $\boldsymbol{f} \in \overline{\mathcal{P}}$. Finally, since $H(\cdot)$ is continuous and $\overline{\mathcal{P}}$ is compact, there exists $\boldsymbol{f}^* \in \overline{\mathcal{P}}$ such that $H(\boldsymbol{f}^*) = C^*$. This means that $C^* = H(\boldsymbol{f}^*) \le G^*$ and combining with $G^* \le C^*$, the result of part (1) follows.

The above argument can be repeated while restricting to the domains $\overline{\mathcal{P}}_0$ and $\overline{\mathcal{P}}_1$ to establish part (2). The claim then follows.

Claim 2:

$$C^* \le \max(C^*_{(0)}, C^*_{(1)}).$$

First observe that for any $\boldsymbol{\omega} \in \mathbb{R}^{D_1}$ and $\delta \in \mathbb{R}$:

$$\begin{aligned} G(\boldsymbol{\omega}, \delta) &= \frac{\sum_{i=1}^{n} c_i \exp\left(\boldsymbol{\omega}^\top \mathbf{z}_i + \delta \cdot b_i\right)}{\sum_{j=1}^{n} \exp\left(\boldsymbol{\omega}^\top \mathbf{z}_j + \delta \cdot b_j\right)} \\ &= G_0(\boldsymbol{\omega}) \cdot \frac{b_0(\boldsymbol{\omega})}{b_0(\boldsymbol{\omega}) + e^{\delta} \cdot b_1(\boldsymbol{\omega})} + G_1(\boldsymbol{\omega}) \cdot \frac{e^{\delta} \cdot b_1(\boldsymbol{\omega})}{b_0(\boldsymbol{\omega}) + e^{\delta} \cdot b_1(\boldsymbol{\omega})}. \end{aligned}$$

That is, $G(\boldsymbol{\omega}, \delta)$ is a convex combination of $G_0(\boldsymbol{\omega})$ and $G_1(\boldsymbol{\omega})$ and consequently we have

$$\forall \boldsymbol{\omega} \in \mathbb{R}^{D_1}, \forall \delta \in \mathbb{R} \quad G(\boldsymbol{\omega}, \delta) \leq \max\left(G_0(\boldsymbol{\omega}), G_1(\boldsymbol{\omega})\right) \leq \max(C^*_{(0)}, C^*_{(1)}),$$

where the last inequality follows from the definition of $C^*_{(0)}$ and $C^*_{(1)}$. The claim then follows from the definition of supremum and the fact that $G^* = C^*$ (from Claim 1 above).

Given Claims 1 and 2 above, there are two cases to consider: $C^*_{(1)} \geq C^*_{(0)}$ or $C^*_{(1)} < C^*_{(0)}$. We focus on the case when $C^*_{(1)} \geq C^*_{(0)}$, the other case can be dealt with a symmetric argument:

Case 1: $\mathbf{C^*_{(1)} \geq C^*_{(0)}}$. For each $\boldsymbol{\omega} \in \mathbb{R}^{D_1}$, define the vector $\tilde{\boldsymbol{f}}(\boldsymbol{\omega}) \in [0,1]^n$ as follows:

$$\tilde{f}_j(\boldsymbol{\omega}) = \begin{cases} f_j^{(1)}(\boldsymbol{\omega}) & \text{if } j \in S_1 \\ 0 & \text{otherwise,} \end{cases}$$

where $\boldsymbol{f}^{(1)}(\boldsymbol{\omega}) \in \mathcal{P}_1$ is as defined above in equation (C.11).

Claim 3:

$$\tilde{\boldsymbol{f}}(\boldsymbol{\omega}) \in \overline{\mathcal{P}} \quad \forall\, \boldsymbol{\omega} \in \mathbb{R}^{D_1}.$$

Given any $\boldsymbol{\omega} \in \mathbb{R}^{D_1}$, consider the sequence $\{(\boldsymbol{\omega}, r)\}_{r \in \mathbb{N}} \subset \mathbb{R}^{D_1+1}$. Now for any $i \in S_0$, it follows that

$$\begin{aligned} \lim_{r \to \infty} f_i(\boldsymbol{\omega}, r) &= \lim_{r \to \infty} \frac{\exp\left(\boldsymbol{\omega}^\top \mathbf{z}_i + r \cdot b_i\right)}{\sum_{\ell=1}^{n} \exp\left(\boldsymbol{\omega}^\top \mathbf{z}_\ell + r \cdot b_\ell\right)} \\ &= \lim_{r \to \infty} \frac{\exp\left(\boldsymbol{\omega}^\top \mathbf{z}_i\right)}{\sum_{j \in S_1} \exp\left(\boldsymbol{\omega}^\top \mathbf{z}_j + r\right) + \sum_{\ell \in S_0} \exp\left(\boldsymbol{\omega}^\top \mathbf{z}_\ell\right)} = 0. \end{aligned}$$

Similarly, for any $j \in S_1$, it follows that

$$\lim_{r \to \infty} f_j(\boldsymbol{\omega}, r) = \lim_{r \to \infty} \frac{\exp\left(\boldsymbol{\omega}^\top \mathbf{z}_j + r \cdot b_j\right)}{\sum_{\ell=1}^{n} \exp\left(\boldsymbol{\omega}^\top \mathbf{z}_\ell + r \cdot b_\ell\right)} = \frac{\exp\left(\boldsymbol{\omega}^\top \mathbf{z}_j\right)}{\sum_{\ell \in S_1} \exp\left(\boldsymbol{\omega}^\top \mathbf{z}_\ell\right)} = f_j^{(1)}(\boldsymbol{\omega}).$$

From the above statements, it follows that

$$\lim_{r \to \infty} \boldsymbol{f}(\boldsymbol{\omega}, r) = \tilde{\boldsymbol{f}}(\boldsymbol{\omega}) \implies \tilde{\boldsymbol{f}}(\boldsymbol{\omega}) \in \overline{\mathcal{P}},$$

where the implication follows since $\overline{\mathcal{P}}$ contains the limit of all convergent sequences in $\mathcal{P}$.

Claim 4:

$$C^* = C^*_{(1)}.$$

From Claim 3, it follows that $\tilde{\boldsymbol{f}}(\boldsymbol{\omega}) \in \overline{\mathcal{P}}$ for all $\boldsymbol{\omega} \in \mathbb{R}^{D_1}$. Then, consider the following:

$$C^* \geq H\left(\tilde{\boldsymbol{f}}(\boldsymbol{\omega})\right) = \sum_{\ell=1}^{n} c_\ell \tilde{f}_\ell(\boldsymbol{\omega}) = \frac{\sum_{j \in S_1} c_j \exp\left(\boldsymbol{\omega}^\top \mathbf{z}_j\right)}{\sum_{\ell \in S_1} \exp\left(\boldsymbol{\omega}^\top \mathbf{z}_\ell\right)} = G_1(\boldsymbol{\omega}).$$

where the first inequality follows from the definition of C^*. Then, it follows that

$$G_1(\boldsymbol{\omega}) \leq C^* \ \ \forall \boldsymbol{\omega} \in \mathbb{R}^{D_1} \implies C^*_{(1)} \leq C^* \ \ (\text{since } C^*_{(1)} \text{ is the supremum}).$$

Combining with Claim 2, it follows that $C^* = C^*_{(1)}$.

Next, let $\boldsymbol{f}^{(1,*)} \in \overline{\mathcal{P}}_1$ denote the optimal solution for problem (P_1). Then, using the arguments given in the proof of Theorem 3.3 above, it follows that there exists $\boldsymbol{\omega}_0^{(1)}, \boldsymbol{\theta}^{(1)} \in \mathbb{R}^{D_1}$ such that $\boldsymbol{f}^{(1,*)} = \boldsymbol{f}^{(1)}(\boldsymbol{\omega}_0^{(1)}, \boldsymbol{\theta}^{(1)})$ where $\boldsymbol{f}^{(1)}(\boldsymbol{\omega}_0^{(1)}, \boldsymbol{\theta}^{(1)})$ are the limiting choice probabilities given by (if $\boldsymbol{f}^{(1,*)} > \mathbf{0}$, then we choose $\boldsymbol{\theta}^{(1)} = \mathbf{0}$):

$$f_j^{(1,*)} = f_j^{(1)}(\boldsymbol{\omega}_0^{(1)}, \boldsymbol{\theta}^{(1)}) = \lim_{r \to \infty} \frac{\exp\left\{(\boldsymbol{\omega}_0^{(1)} + r \cdot \boldsymbol{\theta}^{(1)})^\top \mathbf{z}_j\right\}}{\sum_{\ell \in S_1} \exp\left\{(\boldsymbol{\omega}_0^{(1)} + r \cdot \boldsymbol{\theta}^{(1)})^\top \mathbf{z}_\ell\right\}} \quad \forall\, j \in S_1. \tag{C.12}$$

Define $\boldsymbol{\omega}_0 = \boldsymbol{\omega}_0^{(1)} \circ (0)$ where recall that $\circ$ denotes the concatenation operator, and $\boldsymbol{\theta} = \boldsymbol{\theta}^{(1)} \circ (\theta_1)$ with $\theta_1 = \sqrt{5} \cdot \|\boldsymbol{\theta}^{(1)}\| \cdot Z_{\max}$, where $Z_{\max}$ is a constant that satisfies $Z_{\max} \geq \|\mathbf{z}_\ell\|$ for all $\ell \in [n]$. Then, we can show the following:

Claim 5:

$$\begin{aligned}
&(1)\ \boldsymbol{\theta}^\top(\mathbf{z}_j \circ b_j - \mathbf{z}_i \circ b_i) > 0 \ \ \forall\, j \in S_1, \forall\, i \in S_0. \\
&(2)\ f_i(\boldsymbol{\omega}_0, \boldsymbol{\theta}) = 0 \ \ \forall\, i \in S_0. \\
&(3)\ f_j(\boldsymbol{\omega}_0, \boldsymbol{\theta}) = f_j^{(1,*)} \ \ \forall\, j \in S_1.
\end{aligned}$$

Proof. We start with part (1). Consider any $j \in S_1$ and any $i \in S_0$, then it follows

$$\begin{aligned}
\boldsymbol{\theta}^\top(\mathbf{z}_j \circ b_j - \mathbf{z}_i \circ b_i) > 0 &\iff \boldsymbol{\theta}^\top(\mathbf{z}_j \circ 1 - \mathbf{z}_i \circ 0) > 0 \\
&\iff {\boldsymbol{\theta}^{(1)}}^\top(\mathbf{z}_j - \mathbf{z}_i) + \theta_1 > 0 \\
&\impliedby -\|\boldsymbol{\theta}^{(1)}\| \cdot \|\mathbf{z}_j - \mathbf{z}_i\| + \theta_1 > 0 \quad \text{(by Cauchy-Schwarz inequality)} \\
&\iff \theta_1 > \|\boldsymbol{\theta}^{(1)}\| \cdot \|\mathbf{z}_j - \mathbf{z}_i\| \\
&\impliedby \theta_1 > \|\boldsymbol{\theta}^{(1)}\| \cdot (\|\mathbf{z}_i\| + \|\mathbf{z}_j\|) \quad \text{(by triangle inequality)} \\
&\impliedby \theta_1 > 2 \cdot \|\boldsymbol{\theta}^{(1)}\| \cdot \max_{1 \leq \ell \leq n} \|\mathbf{z}_\ell\| \\
&\impliedby \theta_1 > 2 \cdot \|\boldsymbol{\theta}^{(1)}\| \cdot Z_{\max}
\end{aligned}$$

and the last inequality is true by the choice of θ_1.

Next, for part (2) consider the following, given any $i \in S_0$:

$$\begin{aligned} 0 \leq f_i(\boldsymbol{\omega}_0, \boldsymbol{\theta}) &= \frac{\exp\left\{(\boldsymbol{\omega}_0 + r \cdot \boldsymbol{\theta})^\top (\mathbf{z}_i \circ 0)\right\}}{\sum_{\ell=1}^n \exp\left\{(\boldsymbol{\omega}_0 + r \cdot \boldsymbol{\theta})^\top (\mathbf{z}_\ell \circ b_\ell)\right\}} \\ &\leq \frac{\exp\left\{(\boldsymbol{\omega}_0 + r \cdot \boldsymbol{\theta})^\top (\mathbf{z}_i \circ 0)\right\}}{\sum_{\ell \in S_1} \exp\left\{(\boldsymbol{\omega}_0 + r \cdot \boldsymbol{\theta})^\top (\mathbf{z}_\ell \circ b_\ell)\right\}} \\ &= \frac{\exp\left({\boldsymbol{\omega}_0^{(1)}}^\top \mathbf{z}_i\right)}{\sum_{\ell \in S_1} \exp\left\{{\boldsymbol{\omega}_0^{(1)}}^\top \mathbf{z}_\ell + r \cdot \boldsymbol{\theta}^\top (\mathbf{z}_\ell \circ b_\ell - \mathbf{z}_i \circ 0)\right\}} \end{aligned}$$

Then using the Squeeze theorem and part (1), it follows that $\lim_{r \to \infty} f_i(\boldsymbol{\omega}_0, \boldsymbol{\theta}) = 0$.

Finally, for part (3), consider the following for any $j \in S_1$:

$$\begin{aligned} & f_j(\boldsymbol{\omega}_0, \boldsymbol{\theta}) \\ &= \lim_{r \to \infty} \frac{\exp\left\{(\boldsymbol{\omega}_0 + r \cdot \boldsymbol{\theta})^\top (\mathbf{z}_j \circ 1)\right\}}{\sum_{\ell=1}^n \exp\left\{(\boldsymbol{\omega}_0 + r \cdot \boldsymbol{\theta})^\top (\mathbf{z}_\ell \circ b_\ell)\right\}} \\ &= \lim_{r \to \infty} \frac{\exp\left\{{\boldsymbol{\omega}_0^{(1)}}^\top \mathbf{z}_j\right\}}{\sum_{\ell \in S_1} \exp\left\{{\boldsymbol{\omega}_0^{(1)}}^\top \mathbf{z}_\ell + r \cdot {\boldsymbol{\theta}^{(1)}}^\top (\mathbf{z}_\ell - \mathbf{z}_j)\right\} + \sum_{i \in S_0} \exp\left\{{\boldsymbol{\omega}_0^{(1)}}^\top \mathbf{z}_i + r \cdot \boldsymbol{\theta}^\top (\mathbf{z}_i \circ b_i - \mathbf{z}_j \circ b_j)\right\}} \\ &= \lim_{r \to \infty} \frac{\exp\left\{{\boldsymbol{\omega}_0^{(1)}}^\top \mathbf{z}_j\right\}}{\sum_{\ell \in S_1} \exp\left\{{\boldsymbol{\omega}_0^{(1)}}^\top \mathbf{z}_\ell + r \cdot {\boldsymbol{\theta}^{(1)}}^\top (\mathbf{z}_\ell - \mathbf{z}_j)\right\} + \sum_{i \in S_0} 0} \quad \text{(from part (1) of Claim 5)} \\ &= f_j^{(1,*)} \quad \text{(from equation (C.12))} \end{aligned}$$

□

Having proved Claim 5, it follows that

$$H\left(\boldsymbol{f}(\boldsymbol{\omega}_0, \boldsymbol{\theta})\right) = \sum_{j \in S_1} c_j f_j^{(1,*)} = C_{(1)}^* = C^*,$$

where the second last equality follows since $\boldsymbol{f}^{(1,*)}$ is an optimal solution to problem (P_1) and the last follows from Claim 4 above. This shows that $\boldsymbol{f}(\boldsymbol{\omega}_0, \boldsymbol{\theta})$ is the optimal solution to the support finding step, which establishes the result.

Case 2: $\mathbf{C}_{(\mathbf{0})}^* > \mathbf{C}_{(\mathbf{1})}^*$. A symmetric argument from above shows $C^* = C_{(0)}^*$ in this case. In addition, if $\boldsymbol{f}^{(0,*)} = \boldsymbol{f}^{(0)}(\boldsymbol{\omega}_0^{(0)}, \boldsymbol{\theta}^{(0)})$ denotes the optimal solution to problem (P_0), where $\boldsymbol{\omega}_0^{(0)}, \boldsymbol{\theta}^{(0)} \in \mathbb{R}^{D_1}$ are computed using the procedure in the proof of Theorem 3.3, then choosing $\boldsymbol{\omega}_0 = \boldsymbol{\omega}_0^{(0)} \circ (0)$ and $\boldsymbol{\theta} = \boldsymbol{\theta}^{(0)} \circ (\theta_0)$ with $\theta_0 = -\sqrt{5} \cdot \|\boldsymbol{\theta}^{(0)}\| \cdot Z_{\max}$, it follows that

$$\begin{aligned} &(1)\ \boldsymbol{\theta}^\top (\mathbf{z}_i \circ b_i - \mathbf{z}_j \circ b_j) > 0 \ \ \forall\, j \in S_1, \forall\, i \in S_0. \\ &(2)\ f_j(\boldsymbol{\omega}_0, \boldsymbol{\theta}) = 0 \ \ \forall\, j \in S_1. \\ &(3)\ f_i(\boldsymbol{\omega}_0, \boldsymbol{\theta}) = f_i^{(0,*)} \ \ \forall\, i \in S_0. \end{aligned}$$

and in addition,

$$H\left(\boldsymbol{f}(\boldsymbol{\omega}_0, \boldsymbol{\theta})\right) = \sum_{i \in S_0} c_i f_i^{(0,*)} = C^*_{(0)} = C^*,$$

so that the optimal solution is $\boldsymbol{f}(\boldsymbol{\omega}_0, \boldsymbol{\theta})$ which is a boundary type that only considers products in S_0.

In the general case, when there is more than one binary feature, the above sequence of arguments shows that $C^* = C^*_{(e^*)}$ for some $e^* \in \mathcal{E}$, so that the optimal solution corresponds to a boundary type that only considers products in the subset S_{e^*}. Further, if $\boldsymbol{f}^{(e^*,*)} = \boldsymbol{f}^{(e^*)}(\boldsymbol{\omega}_0^{(e^*)}, \boldsymbol{\theta}^{(e^*)})$ denotes the optimal solution to problem (P_e), then choosing

$$\boldsymbol{\omega}_0 = \boldsymbol{\omega}_0^{(e^*)} \circ \underbrace{(0, 0, \ldots, 0)}_{D_2 \text{ times}} \quad \text{and} \quad \boldsymbol{\theta} = \boldsymbol{\theta}^{(e^*)} \circ \boldsymbol{\theta}_e \; ; \quad \boldsymbol{\theta}_e = \sqrt{5} \cdot \|\boldsymbol{\theta}^{(e^*)}\| \cdot Z_{\max} \cdot \left(2 \cdot \mathbf{b}_{e^*} - \underbrace{(1, 1, \ldots, 1)}_{D_2 \text{ times}}\right),$$

where $\mathbf{b}_{e^*} \in \{0,1\}^{D_2}$ is the binary feature vector for products in the equivalence class S_{e^*}, it follows that

$$\begin{aligned}
&(1)\ \boldsymbol{\theta}^\top (\mathbf{z}_j \circ b_j - \mathbf{z}_i \circ b_i) > 0 \ \ \forall\, j \in S_{e^*}, \forall\, i \in [n] \setminus S_{e^*}. \\
&(2)\ f_i(\boldsymbol{\omega}_0, \boldsymbol{\theta}) = 0 \ \ \forall\, i \notin S_{e^*}. \\
&(3)\ f_j(\boldsymbol{\omega}_0, \boldsymbol{\theta}) = f_j^{(e^*,*)} \ \ \forall\, j \in S_{e^*},
\end{aligned}$$

which implies that

$$H\left(\boldsymbol{f}(\boldsymbol{\omega}_0, \boldsymbol{\theta})\right) = \sum_{j \in S_{e^*}} c_j f_j^{(e^*,*)} = C^*_{(e^*)} = C^*,$$

so that the optimal solution to the support finding step is $\boldsymbol{f}(\boldsymbol{\omega}_0, \boldsymbol{\theta})$ which is a boundary type that only considers products in S_{e^*}.

C.6 Algorithm implementation details

We discuss here our approach for solving the support finding step in the CG algorithm in each of our numerical studies.

C.6.1 Synthetic data

For the experiments in Sections 3.5 and 3.7—since the goal is to recover the underlying mixing distribution—we employ the standard off-the-shelf BFGS solver [NW06] to compute a candidate solution for the support finding step, which was enough to obtain an improving objective value in each iteration. Since the subproblem in the support finding step is non-convex, we run the BFGS solver from 20 different (randomly chosen) starting values to ensure that we sufficiently explore the parameter space, and choose the solution which obtains the best objective.

C.6.2 SUSHI dataset

As described in the main text, there were two kinds of sushi varieties—maki and non-maki, represented using a single binary feature. Let S_{maki} and $S_{\text{non-maki}}$ refer to the two kinds of sushi varieties so that $[n] = S_{\text{maki}} \cup S_{\text{non-maki}}$. Let $\mathbf{z}_i \in \mathbb{R}^4$ denote the remaining (non-binary) features for each sushi variety $i \in [n]$ and let $\mathcal{Z}_{\text{maki}}$ and $\mathcal{Z}_{\text{non-maki}}$ denote the convex polytope w.r.t. to these features for both kinds respectively (similar to the definition $\mathcal{Z}_n$ in the main text). Finally, let $\mathcal{J}_{\text{maki}}$ and $\mathcal{J}_{\text{non-maki}}$ denote the extreme points of the polytopes $\mathcal{Z}_{\text{maki}}$ and $\mathcal{Z}_{\text{non-maki}}$.

We take inspiration from the results (in particular the proofs) of Theorems 3.4 and 3.6 to come up with the heuristic approach in Algorithm 12 for solving the support finding step (refer to equation (3.5) in the main text). In particular, the types $\boldsymbol{f}^{\text{maki}}$ and $\boldsymbol{f}^{\text{non-maki}}$ are constructed according to the arguments in the proof of Theorem 3.6 above.

Algorithm 12 Solving the support finding step for the SUSHI dataset

1: $C_{\text{maki,ext}} \leftarrow \max_{j \in \mathcal{J}_{\text{maki}}} c_j$; $C_{\text{non-maki,ext}} \leftarrow \max_{j \in \mathcal{J}_{\text{non-maki}}} c_j$

2: Let $C_{\text{maki,BFGS}}$ and $\boldsymbol{\omega}_{\text{maki,BFGS}}$ be the best objective and corresponding solution of the following subproblem as returned by the standard BFGS solver

$$\max_{\boldsymbol{\omega} \in \mathbb{R}^4} \sum_{i \in S_{\text{maki}}} c_i \cdot \left(\frac{\exp\left(\boldsymbol{\omega}^\top \mathbf{z}_i\right)}{\sum_{j \in S_{\text{maki}}} \exp\left(\boldsymbol{\omega}^\top \mathbf{z}_j\right)} \right)$$

Similarly, compute $C_{\text{non-maki,BFGS}}$ and $\boldsymbol{\omega}_{\text{non-maki,BFGS}}$.

3: $C_{\text{maki}} \leftarrow \max\left(C_{\text{maki,BFGS}}, C_{\text{maki,ext}}\right)$ and $C_{\text{non-maki}} \leftarrow \max\left(C_{\text{non-maki,BFGS}}, C_{\text{non-maki,ext}}\right)$

4: If $C_{\text{maki}} \geq C_{\text{non-maki}}$, then output type $\boldsymbol{f}^{\text{maki}}$ that only considers maki sushi varieties, otherwise output type $\boldsymbol{f}^{\text{non-maki}}$ that only considers `non-maki` sushi varieties

C.6.3 IRI dataset

Since we had more than one offer-set (with varying prices), we could not utilize the heuristic approach outlined in Algorithm 12 above for solving the support finding step. So instead, we just used the BFGS solver to determine an approximate solution. The choice probabilities $\boldsymbol{f}^{\text{BFGS}}$ obtained using the BFGS solver contained entries which were very "small" ($< 10^{-5}$), indicating the (possible) presence of boundary types. Motivated by this, we considered the heuristic approach outlined in Algorithm 13 to determine whether a recovered type was a boundary type (which we base on the proof of Theorem 3.3 above). In particular, let $\mathbf{z}_i = \mathbf{e}_i \circ p_i$ denote the feature vector for product $i \in [n]$, where $e_{ij} = \mathbf{1}[j = i]$ and $\mathbf{1}[\cdot]$ is the indicator variable, and p_i is the price of product i. In the algorithm, similar to the proof

earlier, we assume that the universe of products is expanded so that the same product with different prices in two offer-sets is indexed as two distinct products.

Algorithm 13 Solving the support finding step for the IRI dataset

1: $\boldsymbol{f}^{\text{BFGS}}, \boldsymbol{\omega}_{\text{BFGS}} \leftarrow$ choice probabilities and logit parameter returned by BFGS solver

2: For each offer-set S_t, let $i_t \leftarrow \arg\max_{j \in S_t} f_{jt}^{\text{BFGS}}$

3: For each offer-set S_t, let $\text{Pairs}_t \leftarrow \{(i_t, j) \mid j \in S_t \setminus \{i_t\} \text{ and } \log\left(\frac{f_{i_t t}^{\text{BFGS}}}{f_{jt}^{\text{BFGS}}}\right) < 10^5\}$ and similarly $\overline{\text{Pairs}}_t \leftarrow \{(i_t, j) \mid j \in S_t \setminus \{i_t\} \text{ and } \log\left(\frac{f_{i_t t}^{\text{BFGS}}}{f_{jt}^{\text{BFGS}}}\right) \geq 10^5\}$

4: Let $\text{Pairs} \leftarrow \cup_{t=1}^{T} \text{Pairs}_t$ and $\overline{\text{Pairs}} \leftarrow \cup_{t=1}^{T} \overline{\text{Pairs}}_t$

5: Let $\boldsymbol{\theta}$ (normalized to unit norm) be the solution of the following linear program (LP):

$$\max_{\boldsymbol{\omega} \in \mathbb{R}^{11}} \sum_{(\bar{i}, \bar{j}) \in \overline{\text{Pairs}}} \boldsymbol{\omega}^\top (\mathbf{z}_{\bar{i}} - \mathbf{z}_{\bar{j}})$$

$$\text{s.t. } \boldsymbol{\omega}^\top (\mathbf{z}_i - \mathbf{z}_j) = 0 \ \ \forall\, (i,j) \in \text{Pairs}; \text{ and } \boldsymbol{\omega}^\top (\mathbf{z}_{\bar{i}} - \mathbf{z}_{\bar{j}}) \geq 0 \ \ \forall\, (\bar{i}, \bar{j}) \in \overline{\text{Pairs}}$$

6: Let $\boldsymbol{\omega}_0 \leftarrow \boldsymbol{\omega}_{\text{BFGS}} - \|\boldsymbol{\omega}_{\text{BFGS}}\| \cdot \boldsymbol{\theta}$

7: Compute $\boldsymbol{f}(\boldsymbol{\omega}_0, \boldsymbol{\theta})$ as the limiting choice probabilities defined in Theorem 3.3

8: If $\left\langle \nabla \mathsf{loss}(\mathbf{g}^{(k-1)}), \boldsymbol{f}(\boldsymbol{\omega}_0, \boldsymbol{\theta}) \right\rangle < \left\langle \nabla \mathsf{loss}(\mathbf{g}^{(k-1)}), \boldsymbol{f}^{\text{BFGS}} \right\rangle$, then output boundary type $\boldsymbol{f}(\boldsymbol{\omega}_0, \boldsymbol{\theta})$, otherwise output non-boundary type $\boldsymbol{f}^{\text{BFGS}}$

We used Gurobi Optimizer version 6.5.1 to solve the LP in Step 5 above.

Bibliography

[AGH+14] Animashree Anandkumar, Rong Ge, Daniel Hsu, Sham M Kakade, and Matus Telgarsky. Tensor decompositions for learning latent variable models. *Journal of Machine Learning Research*, 15(Aug):2773–2832, 2014.

[AM05] Dimitris Achlioptas and Frank McSherry. On spectral learning of mixtures of distributions. In *Learning Theory*, pages 458–469. Springer, 2005.

[AMMIL12] Yaser S. Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin. *Learning From Data*. AMLBook, 2012.

[Ass70] Henry Assael. Segmenting markets by group purchasing behavior: an application of the aid technique. *Journal of Marketing Research*, 7(2):153–158, 1970.

[Bac13] Francis Bach. Learning with submodular functions: A convex optimization perspective. *Foundations and Trends® in Machine Learning*, 6(2–3):145–373, 2013.

[Bha97] Chandra R Bhat. An endogenous segmentation mode choice model with an application to intercity travel. *Transportation Science*, 31(1):34–48, 1997.

[BJS+16] Yuri M. Brovman, Marie Jacob, Natraj Srinivasan, Stephen Neola, Daniel Galron, Ryan Snyder, and Paul Wang. Optimizing similar item recommendations in a semi-structured marketplace to maximize conversion. In *Proceedings of the 10th ACM Conference on Recommender systems*, RecSys '16, pages 199–202. ACM, 2016.

[BK07] Robert M Bell and Yehuda Koren. Lessons from the netflix prize challenge. *Acm Sigkdd Explorations Newsletter*, 9(2):75–79, 2007.

[BKM08] Bart J Bronnenberg, Michael W Kruger, and Carl F Mela. Database paper-the iri marketing data set. *Marketing Science*, 27(4):745–748, 2008.

[BLMK12] S Derin Babacan, Martin Luessi, Rafael Molina, and Aggelos K Katsaggelos. Sparse bayesian methods for low-rank matrix estimation. *IEEE Transactions on Signal Processing*, 60(8):3964–3977, 2012.

[BLP95] Steven Berry, James Levinsohn, and Ariel Pakes. Automobile prices in market equilibrium. *Econometrica*, 63(4):841–890, 1995.

[Bol01] Béla Bollobás. *Random Graphs*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, second edition, 2001.

[BSL92] Dankmar Bohning, Peter Schlattmann, and Bruce Lindsay. Computer-assisted analysis of mixtures (ca man): statistical algorithms. *Biometrics*, 48(1):283–303, 1992.

[Cla10] Kenneth L Clarkson. Coresets, sparse greedy approximation, and the frank-wolfe algorithm. *ACM Transactions on Algorithms (TALG)*, 6(4):63:1–63:30, 2010.

[CLZ13] Xi Chen, Qihang Lin, and Dengyong Zhou. Optimistic knowledge gradient policy for optimal budget allocation in crowdsourcing. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 64–72, 2013.

[CM02] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[DDCM12] Djellel Eddine Difallah, Gianluca Demartini, and Philippe Cudré-Mauroux. Mechanical cheat: Spamming schemes and adversarial techniques on crowdsourcing platforms. In *Proceedings of the First International Workshop on Crowdsourcing Web Search*, pages 26–30, 2012.

[DDKR13] Nilesh Dalvi, Anirban Dasgupta, Ravi Kumar, and Vibhor Rastogi. Aggregating crowdsourced binary ratings. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 285–294, 2013.

[DGK04] Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. Kernel k-means: spectral clustering and normalized cuts. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 551–556. ACM, 2004.

[dHINM04] Michiel JL de Hoon, Seiya Imoto, John Nolan, and Satoru Miyano. Open source clustering software. *Bioinformatics*, 20(9):1453–1454, 2004.

[DHSC10] Julie S Downs, Mandy B Holbrook, Steve Sheng, and Lorrie Faith Cranor. Are your participants gaming the system?: Screening mechanical turk workers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2399–2402. ACM, 2010.

[DKNS01] Cynthia Dwork, Ravi Kumar, Moni Naor, and Dandapani Sivakumar. Rank aggregation methods for the web. In *Proceedings of the 10th International Conference on World Wide Web*, pages 613–622, 2001.

[DM09] George Danezis and Prateek Mittal. Sybilinfer: Detecting sybil nodes using social networks. In *Proceedings of the Network and Distributed System Security Symposium*, pages 1–15, 2009.

[DMM94] Wayne S DeSarbo, Ajay K Manrai, and Lalita A Manrai. Latent class multidimensional scaling. a review of recent developments in the marketing and psychometric literature. *Advanced Methods of Marketing Research, R. Bagozzi (Ed.), Blackwell Pub*, pages 190–222, 1994.

[DS79] Alexander Philip Dawid and Allan M Skene. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Applied Statistics*, 28(1):20–28, 1979.

[FD18] Long Feng and Lee H Dicker. Approximate nonparametric maximum likelihood for mixture models: A convex optimization approach to fitting arbitrary multivariate mixing distributions. *Computational Statistics & Data Analysis*, 122:80–91, 2018.

[FKK+11] Michael J Franklin, Donald Kossmann, Tim Kraska, Sukriti Ramesh, and Reynold Xin. Crowddb: Answering queries with crowdsourcing. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, pages 61–72. ACM, 2011.

[FKRB11] Jeremy T Fox, Kyoo il Kim, Stephen P Ryan, and Patrick Bajari. A simple estimator for the distribution of random coefficients. *Quantitative Economics*, 2(3):381–418, 2011.

[FR02] Chris Fraley and Adrian E Raftery. Model-based clustering, discriminant analysis, and density estimation. *Journal of the American statistical Association*, 97(458):611–631, 2002.

[FW56] Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110, 1956.

[GC94] Sachin Gupta and Pradeep K Chintagunta. On using demographic variables to determine segment membership in logit mixture models. *Journal of Marketing Research*, 31(1):128–136, 1994.

[GH15] Dan Garber and Elad Hazan. Faster rates for the frank-wolfe method over strongly-convex sets. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 541–549, 2015.

[GKDD15] Ujwal Gadiraju, Ricardo Kawase, Stefan Dietze, and Gianluca Demartini. Understanding malicious behavior in crowdsourcing platforms: The case of online surveys. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 1631–1640, 2015.

[GKM11] Arpita Ghosh, Satyen Kale, and Preston McAfee. Who moderates the moderators? crowdsourcing abuse detection in user-generated content. In *Proceedings of the 12th ACM Conference on Electronic Commerce*, pages 167–176. ACM, 2011.

[GKT95] Ram Gnanadesikan, Jon R Kettenring, and Shiao Li Tsao. Weighting and selection of variables for cluster analysis. *Journal of Classification*, 12(1):113–136, 1995.

[GM86] Jacques Guélat and Patrice Marcotte. Some comments on wolfe's 'away step'. *Mathematical Programming*, 35(1):110–119, 1986.

[GR10] Dilan Görür and Carl Edward Rasmussen. Dirichlet process gaussian mixture models: Choice of the base distribution. *Journal of Computer Science and Technology*, 25(4):653–664, 2010.

[GZ16] Chao Gao and Dengyong Zhou. Minimax optimal convergence rates for estimating ground truth from crowdsourced labels. arXiv preprint arXiv:1310.5764v6, 2016. https://arxiv.org/abs/1310.5764.

[Hau14] John R Hauser. Consideration-set heuristics. *Journal of Business Research*, 67(8):1688–1699, 2014.

[HBKVH13] Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. Learning whom to trust with mace. In *Proceedings of NAACL-HLT*, pages 1120–1130, 2013.

[HJN15] Zaid Harchaoui, Anatoli Juditsky, and Arkadi Nemirovski. Conditional gradient algorithms for norm-regularized smooth convex optimization. *Mathematical Programming*, 152(1-2):75–112, 2015.

[HK13] Daniel Hsu and Sham M Kakade. Learning mixtures of spherical gaussians: moment methods and spectral decompositions. In *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*, pages 11–20. ACM, 2013.

[HKBR99] Jonathan L Herlocker, Joseph A Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 230–237. ACM, 1999.

[HLLT03] Nicholas JA Harvey, Richard E Ladner, László Lovász, and Tami Tamir. Semi-matchings for bipartite graphs and load balancing. In *Workshop on Algorithms and Data Structures*, pages 294–306. Springer, 2003.

[HSS08] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy)*, pages 11–15, 2008.

[Hun04] David R Hunter. MM algorithms for generalized bradley-terry models. *The Annals of Statistics*, 32(1):384–406, 2004.

[IPW10] Panagiotis G Ipeirotis, Foster Provost, and Jing Wang. Quality management on amazon mechanical turk. In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, pages 64–67. ACM, 2010.

[IR10] Alexander Ilin and Tapani Raiko. Practical approaches to principal component analysis in the presence of missing values. *Journal of Machine Learning Research*, 11(Jul):1957–2000, 2010.

[Jag11] Martin Jaggi. *Sparse convex optimization methods for machine learning*. PhD thesis, ETH Zürich, 2011.

[Jag13] Martin Jaggi. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 427–435, 2013.

[Jai10] Anil K Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.

[Jam17] Jonathan James. MM algorithm for general mixed multinomial logit models. *Journal of Applied Econometrics*, 32(4):841–857, 2017.

[JR16] Srikanth Jagabathula and Paat Rusmevichientong. A nonparametric joint assortment and price choice model. *Management Science*, 63(9):3128–3145, 2016.

[JR17] Srikanth Jagabathula and Paat Rusmevichientong. The limit of rationality in choice modeling: Formulation, computation, and implications. *To appear in Management Science*, 2017.

[JS10] Martin Jaggi and Marek Sulovsk. A simple algorithm for nuclear norm regularized problems. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 471–478, 2010.

[JSV17] Srikanth Jagabathula, Lakshminarayanan Subramanian, and Ashwin Venkataraman. Identifying unreliable and adversarial workers in crowdsourced labeling tasks. *Journal of Machine Learning Research*, 18(93):1–67, 2017.

[JSV18a] Srikanth Jagabathula, Lakshminarayanan Subramanian, and Ashwin Venkataraman. A conditional gradient approach for nonparametric estimation of mixing distributions, 2018. Draft available at https://ssrn.com/abstract=3029881.

[JSV18b] Srikanth Jagabathula, Lakshminarayanan Subramanian, and Ashwin Venkataraman. A model-based embedding technique for segmenting customers. *To appear in Operations Research*, 2018. Draft available at https://ssrn.com/abstract=2696161.

[JTFF14] Armand Joulin, Kevin Tang, and Li Fei-Fei. Efficient image and video co-localization with frank-wolfe algorithm. In *Computer Vision – ECCV 2014*, pages 253–268. Springer, 2014.

[JZ09] Wenhua Jiang and Cun-Hui Zhang. General maximum likelihood empirical bayes estimation of normal means. *The Annals of Statistics*, 37(4):1647–1684, 2009.

[KA18] Matthaeus Kleindessner and Pranjal Awasthi. Crowdsourcing with arbitrary adversaries. In *Proceedings of the 35th International Conference on Machine Learning (ICML-18)*, pages 2713–2722, 2018.

[Kam88] Wagner A Kamakura. A least squares procedure for benefit segmentation with conjoint experiments. *Journal of Marketing Research*, 25(2):157–67, 1988.

[KBV+09] Yehuda Koren, Robert Bell, Chris Volinsky, et al. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[KCS08] Aniket Kittur, Ed H Chi, and Bongwon Suh. Crowdsourcing user studies with mechanical turk. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 453–456. ACM, 2008.

[KKA05] Toshihiro Kamishima, Hideto Kazawa, and Shotaro Akaho. Supervised ordering — an empirical survey. In *Fifth IEEE International Conference on Data Mining*, pages 673–676. IEEE, 2005.

[KKH15] Ece Kamar, Ashish Kapoor, and Eric Horvitz. Identifying and accounting for task-dependent bias in crowdsourcing. In *Third AAAI Conference on Human Computation and Crowdsourcing*, pages 92–101, 2015.

[KKL96] Wagner A Kamakura, Byung-Do Kim, and Jonathan Lee. Modeling preference and structural heterogeneity in consumer choice. *Marketing Science*, 15(2):152–172, 1996.

[KLJS15] Rahul G Krishnan, Simon Lacoste-Julien, and David Sontag. Barrier frank-wolfe for marginal inference. In *Advances in Neural Information Processing Systems 28*, pages 532–540. 2015.

[KO16] Ashish Khetan and Sewoong Oh. Achieving budget-optimality with adaptive schemes in crowdsourcing. In *Advances in Neural Information Processing Systems 29*, pages 4844–4852, 2016.

[KOS11] David R Karger, Sewoong Oh, and Devavrat Shah. Iterative learning for reliable crowdsourcing systems. In *Advances in Neural Information Processing Systems 24*, pages 1953–1961, 2011.

[KOS13] David R Karger, Sewoong Oh, and Devavrat Shah. Efficient crowdsourcing for multi-class labeling. In *Proceedings of ACM SIGMETRICS*, pages 81–92. ACM, 2013.

[KOS14] David R Karger, Sewoong Oh, and Devavrat Shah. Budget-optimal task allocation for reliable crowdsourcing systems. *Operations Research*, 62(1):1–24, 2014.

[KR89] Wagner A Kamakura and Gary Russell. A probabilistic choice model for market segmentation and elasticity structure. *Journal of Marketing Research*, 26(4):379–390, 1989.

[KSV05] Ravindran Kannan, Hadi Salmasian, and Santosh Vempala. The spectral method for general mixture models. In *Learning Theory*, pages 444–457. Springer, 2005.

[KW56] Jack Kiefer and Jacob Wolfowitz. Consistency of the maximum likelihood estimator in the presence of infinitely many incidental parameters. *The Annals of Mathematical Statistics*, 27(4):887–906, 1956.

[Lai78] Nan Laird. Nonparametric maximum likelihood estimation of a mixing distribution. *Journal of the American Statistical Association*, 73(364):805–811, 1978.

[LEHB10] John Le, Andy Edmonds, Vaughn Hester, and Lukas Biewald. Ensuring quality in crowdsourced search relevance evaluation: The effects of training question distribution. In *Proc. ACM SIGIR Workshop on Crowdsourcing for Search Evaluation*, pages 21–26. ACM, 2010.

[LFJ04] Martin HC Law, Mario AT Figueiredo, and Anil K Jain. Simultaneous feature selection and clustering using mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1154–1166, 2004.

[Lin83] Bruce G Lindsay. The geometry of mixture likelihoods: A general theory. *The Annals of Statistics*, 11(1):86–94, 1983.

[Liu09] Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3):225–331, 2009.

[LJJ15] Simon Lacoste-Julien and Martin Jaggi. On the global linear convergence of frank-wolfe optimization variants. In *Advances in Neural Information Processing Systems 28*, pages 496–504, 2015.

[LLLG95] Eric Lapersonne, Gilles Laurent, and Jean-Jacques Le Goff. Consideration sets of size one: An empirical investigation of automobile purchases. *International Journal of Research in Marketing*, 12(1):55–66, 1995.

[LPI12] Qiang Liu, Jian Peng, and Alex Ihler. Variational inference for crowdsourcing. In *Advances in Neural Information Processing Systems 25*, pages 701–709, 2012.

[LSKC13] Jovian Lin, Kazunari Sugiyama, Min-Yen Kan, and Tat-Seng Chua. Addressing cold-start in app recommendation: latent user models constructed from twitter followers. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 283–292. ACM, 2013.

[LY14] Hongwei Li and Bin Yu. Error rate bounds and iterative weighted majority voting for crowdsourcing. arXiv preprint arXiv:1411.4086, 2014. http://arxiv.org/abs/1411.4086.

[LZF14] Hongwei Li, Bo Zhao, and Ariel Fuxman. The wisdom of minority: Discovering and targeting the right group of workers for crowdsourcing. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 165–176, 2014.

[MDMBVZ14] Isabel Méndez-Díaz, Juan José Miranda-Bront, Gustavo Vulcano, and Paula Zabala. A branch-and-cut algorithm for the latent-class logit assortment problem. *Discrete Applied Mathematics*, 164:246–263, 2014.

[MHT10] Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. *Journal of Machine Learning Research*, 11(Aug):2287–2322, 2010.

[MJ81] Douglas L Maclachlan and Johny K Johansson. Market segmentation with multivariate aid. *Journal of Marketing*, 45(1):74–84, 1981.

[MKKSM13] Arash Molavi Kakhki, Chloe Kliman-Silver, and Alan Mislove. Iolaus: Securing online content rating systems. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 919–930, 2013.

[MLG12] Arjun Mukherjee, Bing Liu, and Natalie Glance. Spotting fake reviewer groups in consumer reviews. In *Proceedings of the 21st International Conference on World Wide Web*, pages 191–200, 2012.

[MLS+18] Chenglin Miao, Qi Li, Lu Su, Mengdi Huai, Wenjun Jiang, and Jing Gao. Attack under disguise: An intelligent data poisoning attack mechanism in crowdsourcing. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 13–22, 2018.

[MML+11] Marti Motoyama, Damon McCoy, Kirill Levchenko, Stefan Savage, and Geoffrey M Voelker. Dirty jobs: The role of freelance labor in web service abuse. In *20th USENIX Security Symposium (USENIX Security 11)*, pages 14–29. USENIX Association, 2011.

[MP00] Geoffrey McLachlan and David Peel. *Finite mixture models*. John Wiley & Sons, 2000.

[MS07] Andriy Mnih and Ruslan Salakhutdinov. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems 20*, pages 1257–1264, 2007.

[MSF+14] Barzan Mozafari, Purna Sarkar, Michael Franklin, Michael Jordan, and Samuel Madden. Scaling up crowd-sourcing to very large datasets: A case for active learning. *Proceedings of the VLDB Endowment*, 8(2):125–136, 2014.

[MT00] Daniel McFadden and Kenneth Train. Mixed MNL models for discrete response. *Journal of Applied Econometrics*, 15(5):447–470, 2000.

[NJW01] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*, pages 849–856. 2001.

[NW06] Jorge Nocedal and Stephen J. Wright. *Numerical optimization*. Springer, second edition, 2006.

[Oga87] Kohsuke Ogawa. An approach to simultaneous estimation and segmentation in conjoint analysis. *Marketing Science*, 6(1):66–81, 1987.

[OOSY16] Jungseul Ok, Sewoong Oh, Jinwoo Shin, and Yung Yi. Optimality of belief propagation for crowdsourced classification. In *Proceedings of the 33rd International Conference on Machine Learning (ICML-16)*, pages 535–544, 2016.

[PC09] Seung-Taek Park and Wei Chu. Pairwise preference regression for cold-start recommendation. In *Proceedings of the 3rd ACM Conference on Recommender systems*, pages 21–28. ACM, 2009.

[PT10] Amil Petrin and Kenneth Train. A control function approach to endogeneity in consumer choice models. *Journal of Marketing Research*, 47(1):3–13, 2010.

[RAM05] Peter E Rossi, Greg M Allenby, and Rob McCulloch. *Bayesian statistics and marketing*. John Wiley & Sons, 2005.

[Ras00] Carl Edward Rasmussen. The infinite gaussian mixture model. In *Advances in Neural Information Processing Systems 13*, pages 554–560, 2000.

[RD06] Adrian E Raftery and Nema Dean. Variable selection for model-based clustering. *Journal of the American Statistical Association*, 101(473):168–178, 2006.

[RL04] Volker Roth and Tilman Lange. Feature selection in clustering problems. In *Advances in Neural Information Processing Systems 17*, pages 473–480, 2004.

[RM05] Lior Rokach and Oded Maimon. Clustering methods. In *Data Mining and Knowledge Discovery Handbook*, pages 321–352. Springer, 2005.

[Rob50] Herbert Robbins. A generalization of the method of maximum likelihood-estimating a mixing distribution. *The Annals of Mathematical Statistics*, 21(2):314–315, 1950.

[RPGMP12] Aditya Ramesh, Aditya Parameswaran, Hector Garcia-Molina, and Neoklis Polyzotis. Identifying reliable workers swiftly. Technical report, 2012.

[RT98] David Revelt and Kenneth Train. Mixed logit with repeated choices: households' choices of appliance efficiency level. *Review of economics and statistics*, 80(4):647–657, 1998.

[RY12] Vikas C Raykar and Shipeng Yu. Eliminating spammers and ranking annotators for crowdsourced labeling tasks. *Journal of Machine Learning Research*, 13(Feb):491–518, 2012.

[RYZ+10] Vikas C Raykar, Shipeng Yu, Linda H Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. Learning from crowds. *Journal of Machine Learning Research*, 11(Apr):1297–1322, 2010.

[SFB+95] Padhraic Smyth, Usama Fayyad, Michael Burl, Pietro Perona, and Pierre Baldi. Inferring ground truth from subjective labelling of venus images. In *Advances in Neural Information Processing Systems 8*, pages 1085–1092, 1995.

[SG02] Alexander Strehl and Joydeep Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3(Dec):583–617, 2002.

[SG11] Guy Shani and Asela Gunawardana. Evaluating recommendation systems. In *Recommender Systems Handbook*, pages 257–297. Springer, 2011.

[SM00] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[SM08] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th International Conference on Machine Learning (ICML-08)*, pages 880–887. ACM, 2008.

[Smi56] Wendell R Smith. Product differentiation and market segmentation as alternative marketing strategies. *Journal of Marketing*, 21(1):3–8, 1956.

[SOJN08] Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y Ng. Cheap and fast—but is it good?: Evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 254–263. Association for Computational Linguistics, 2008.

[SPI08] Victor S Sheng, Foster Provost, and Panagiotis G Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 614–622. ACM, 2008.

[SPUP02] Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 253–260. ACM, 2002.

[SSB+14] Suvash Sedhain, Scott Sanner, Darius Braziunas, Lexing Xie, and Jordan Christensen. Social collaborative filtering for cold-start recommendations. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 345–348. ACM, 2014.

[SSSZ10] Shai Shalev-Shwartz, Nathan Srebro, and Tong Zhang. Trading accuracy for sparsity in optimization problems with sparsity constraints. *SIAM Journal on Optimization*, 20(6):2807–2832, 2010.

[Teh11] Yee Whye Teh. Dirichlet process. In *Encyclopedia of machine learning*, pages 280–287. Springer, 2011.

[TF13] Vincent YF Tan and Cedric Fevotte. Automatic relevance determination in nonnegative matrix factorization with the/spl beta/-divergence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(7):1592–1605, 2013.

[TL11] Wei Tang and Matthew Lease. Semi-supervised consensus labeling for crowdsourcing. In *Proc. ACM SIGIR Workshop on Crowdsourcing for Information Retrieval*, pages 1–6, 2011.

[TLSC11] Nguyen Tran, Jinyang Li, Lakshminarayanan Subramanian, and Sherman SM Chow. Optimal sybil-resilient node admission control. In *Proceedings of IEEE INFOCOM*, pages 3218–3226. IEEE, 2011.

[TMLS09] Nguyen Tran, Bonan Min, Jinyang Li, and Lakshminarayanan Subramanian. Sybil-resilient online content voting. In *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation*, pages 15–28. USENIX Association, 2009.

[TPNT09] Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk. Scalable collaborative filtering approaches for large recommender systems. *Journal of Machine Learning Research*, 10(Mar):623–656, 2009.

[Tra08] Kenneth E Train. EM algorithms for nonparametric estimation of mixing distributions. *Journal of Choice Modelling*, 1(1):40–69, 2008.

[Tra09] Kenneth E Train. *Discrete choice methods with simulation*. Cambridge university press, 2009.

[Vad12] Salil P Vadhan. Pseudorandomness. *Foundations and Trends® in Theoretical Computer Science*, 7(1–3):1–336, 2012.

[VdVE11] Jeroen Vuurens, Arjen P de Vries, and Carsten Eickhoff. How much spam can you take? an analysis of crowdsourcing results to increase accuracy. In *Proc. ACM SIGIR Workshop on Crowdsourcing for Information Retrieval*, pages 21–26, 2011.

[VMG+12] Bimal Viswanath, Mainack Mondal, Krishna P Gummadi, Alan Mislove, and Ansley Post. Canal: Scaling social network-based sybil tolerance schemes. In *Proceedings of the 7th ACM European Conference on Computer Systems*, pages 309–322. ACM, 2012.

[VPGM10] Bimal Viswanath, Ansley Post, Krishna P Gummadi, and Alan Mislove. An analysis of social network-based sybil defenses. *ACM SIGCOMM Computer Communication Review*, 40:363–374, 2010.

[Wan10] Junhui Wang. Consistent selection of the number of clusters via crossvalidation. *Biometrika*, 97(4):893–904, 2010.

[WBBP10] Peter Welinder, Steve Branson, Serge Belongie, and Pietro Perona. The multidimensional wisdom of crowds. In *Advances in Neural Information Processing Systems 23*, pages 2424–2432, 2010.

[WD94] Michel Wedel and Wayne S DeSarbo. A review of recent developments in latent class regression models. *Advanced Methods of Marketing Research, R. Bagozzi (Ed.), Blackwell Pub*, pages 352–388, 1994.

[WJ08] Martin J Wainwright and Michael I Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008.

[WK89] Michel Wedel and Cor Kistemaker. Consumer benefit segmentation using clusterwise linear regression. *International Journal of Research in Marketing*, 6(1):45–59, 1989.

[WK00] Michel Wedel and Wagner A Kamakura. *Market segmentation: Conceptual and methodological foundations*. Springer Science & Business Media, second edition, 2000.

[WRW+09] Jacob Whitehill, Paul Ruvolo, Tingfan Wu, Jacob Bergsma, and Javier Movellan. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Advances in Neural Information Processing Systems 22*, pages 2035–2043, 2009.

[WS89] Michel Wedel and Jan-Benedict EM Steenkamp. A fuzzy clusterwise regression approach to benefit segmentation. *International Journal of Research in Marketing*, 6(4):241–258, 1989.

[WSD+16] Yu-Xiang Wang, Veeranjaneyulu Sadhanala, Wei Dai, Willie Neiswanger, Suvrit Sra, and Eric Xing. Parallel and distributed block-coordinate frank-wolfe algorithms. In *Proceedings of the 33rd International Conference on Machine Learning (ICML-16)*, pages 1548–1557, 2016.

[WWW+16] Gang Wang, Bolun Wang, Tianyi Wang, Ana Nika, Haitao Zheng, and Ben Y Zhao. Defending against sybil devices in crowdsourced mapping services. In *Proceedings of the 14th International Conference on Mobile Systems, Applications, and Services*, pages 179–191, 2016.

[WWZ+12] Gang Wang, Christo Wilson, Xiaohan Zhao, Yibo Zhu, Manish Mohanlal, Haitao Zheng, and Ben Y Zhao. Serf and turf: Crowdturfing for fun and profit. In *Proceedings of the 21st International Conference on World Wide Web*, pages 679–688, 2012.

[WWZZ14] Gang Wang, Tianyi Wang, Haitao Zheng, and Ben Y Zhao. Man vs. machine: Practical adversarial detection of malicious crowdsourcing workers. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 239–254. USENIX Association, 2014.

[WY14] Naiyan Wang and Dit-Yan Yeung. Ensemble-based tracking: Aggregating crowdsourced structured time series data. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1107–1115, 2014.

[XW05] Rui Xu and Donald Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005.

[YGKX08] Haifeng Yu, Phillip B Gibbons, Michael Kaminsky, and Feng Xiao. Sybillimit: A near-optimal social network defense against sybil attacks. In *IEEE Symposium on Security and Privacy*, pages 3–17, 2008.

[YH11] Christopher Yau and Chris Holmes. Hierarchical bayesian nonparametric mixture models for clustering with variable relevance determination. *Bayesian Analysis*, 6(2):329–351, 2011.

[YKGF06] Haifeng Yu, Michael Kaminsky, Phillip B Gibbons, and Abraham Flaxman. Sybilguard: Defending against sybil attacks via social networks. *ACM SIGCOMM Computer Communication Review*, 36:267–278, 2006.

[YLLZ17] Dong Yuan, Guoliang Li, Qi Li, and Yudian Zheng. Sybil defense in crowdsourcing platforms. In *Proceedings of the 2017 ACM Conference on Information and Knowledge Management*, pages 1529–1538, 2017.

[Zan69] Willard I Zangwill. *Nonlinear programming: a unified approach*. Prentice-Hall, 1969.

[ZBMP12] Dengyong Zhou, Sumit Basu, Yi Mao, and John C Platt. Learning from the wisdom of crowds by minimax entropy. In *Advances in Neural Information Processing Systems 25*, pages 2195–2203, 2012.

[ZCZJ14] Yuchen Zhang, Xi Chen, Dengyong Zhou, and Michael I Jordan. Spectral methods meet EM: A provably optimal algorithm for crowdsourcing. In *Advances in Neural Information Processing Systems 27*, pages 1260–1268, 2014.

[ZG03] Shi Zhong and Joydeep Ghosh. A unified framework for model-based clustering. *Journal of Machine Learning Research*, 4(Nov):1001–1037, 2003.

[Zha03] Tong Zhang. Sequential greedy approximation for certain convex optimization problems. *IEEE Transactions on Information Theory*, 49(3):682–691, 2003.

[ZLP+15] Dengyong Zhou, Qiang Liu, John C Platt, Christopher Meek, and Nihar B Shah. Regularized minimax conditional entropy for crowdsourcing. arXiv preprint arXiv:1503.07240, 2015. http://arxiv.org/abs/1503.07240.

[ZTZX14] Mi Zhang, Jie Tang, Xuchen Zhang, and Xiangyang Xue. Addressing cold start in recommender systems: A semi-supervised co-training algorithm. In *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 73–82. ACM, 2014.